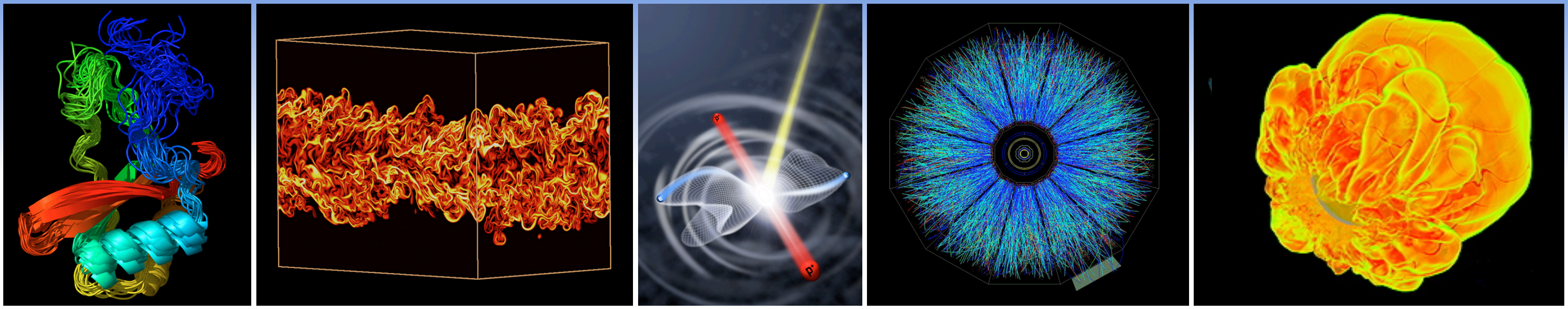


Green Flash: Berkeley Lab Research into Energy Efficient HPC

David Donofrio

John Shalf, Leonid Oliker, Michael Wehner

National Energy Research Supercomputing Center

Lawrence Berkeley National Laboratory

LCI Conference

Pittsburgh, PA, March 10, 2010

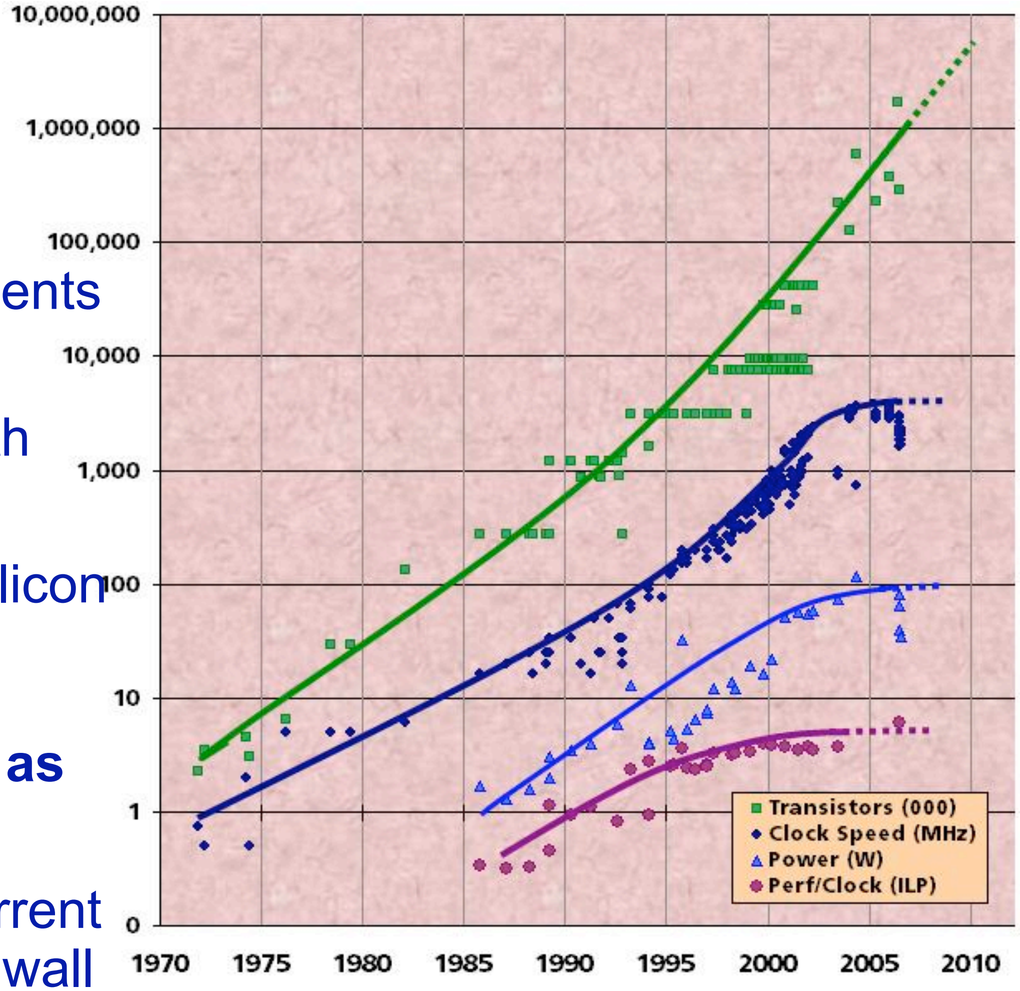
Office of Science
U.S. DEPARTMENT OF ENERGY

NERSC

BERKELEY LAB

New Design Constraint: *POWER*

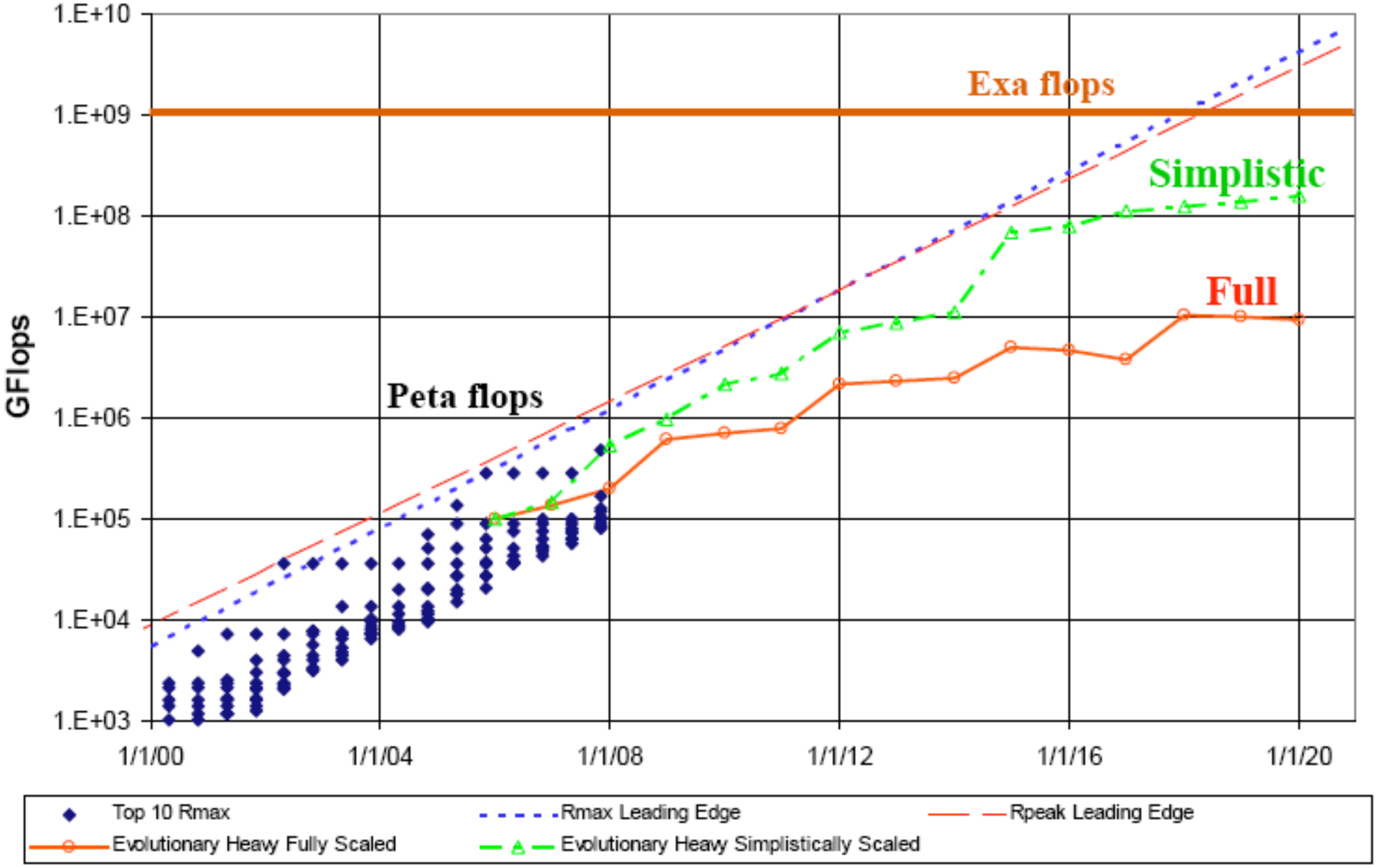
- **Transistors still getting smaller**
  - Moore’s Law is alive and well
- **But Dennard scaling is dead!**
  - No power efficiency improvements with smaller transistors
  - No clock frequency scaling with smaller transistors
  - All “magical improvement of silicon goodness” has ended
- **Cannot continue with business as usual**
  - DARPA study extrapolated current design trends and found brick wall at end of exponential curves



Olukotun et. al.

NERSC
NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

Cannot continue Performance Scaling with Current Approach

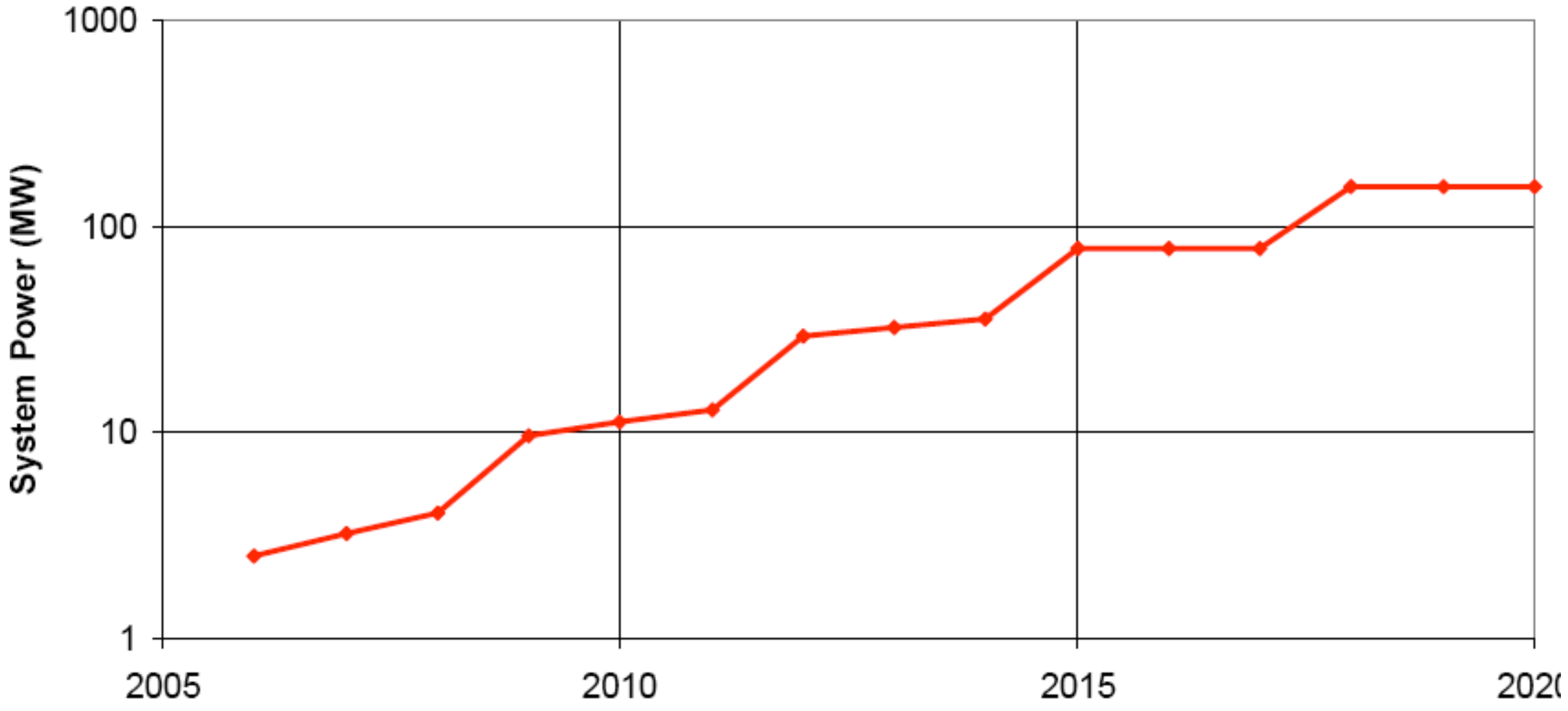


From Peter Kogge, DARPA Exascale Study

Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

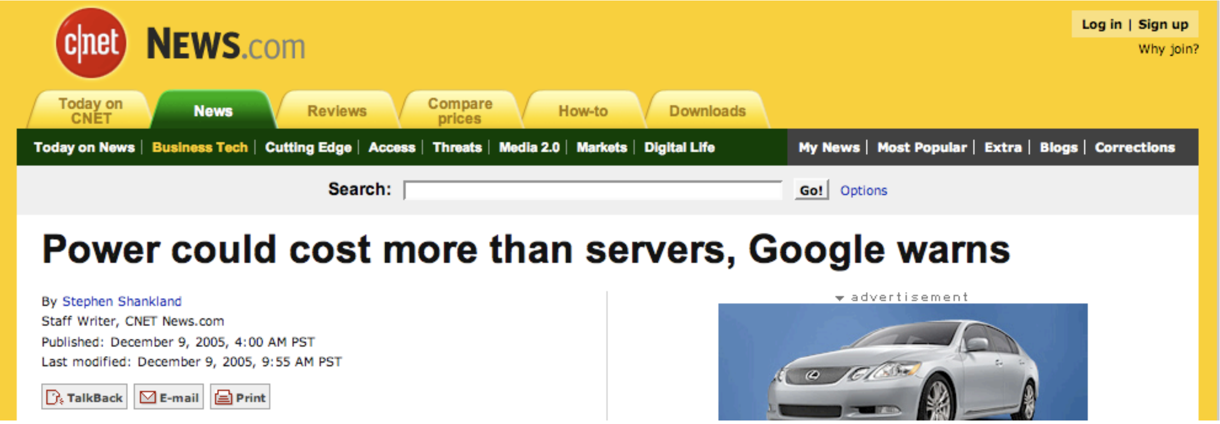
… and the power costs will still be staggering



From Peter Kogge,
DARPA Exascale Study

Power is an Industry Wide Problem

(2% of US power consumption and growing)


cnet NEWS.com
Log in | Sign up
Why join?
Today on CNET
News
Reviews
Compare prices
How-to
Downloads
Today on News | Business Tech | Cutting Edge | Access | Threats | Media 2.0 | Markets | Digital Life
My News | Most Popular | Extra | Blogs | Corrections
Search:
Go!
Options
advertisement
TalkBack
E-mail
Print

Power could cost more than servers, Google warns

By Stephen Shankland
Staff Writer, CNET News.com
Published: December 9, 2005, 4:00 AM PST
Last modified: December 9, 2005, 9:55 AM PST

The New York Times "Hiding in Plain Sight, Google Seeks More Power", by John Markoff, June 14, 2006

New Google Plant in The Dulles, Oregon, from NYT, June 14, 2006

Relocate to Iceland?

The Challenge

How to get 1000x performance without building a nuclear power plant next to my HPC center?

How do you achieve this in 10 years with a finite development budget?

How do you make it "programmable?"

Green Flash: Overview

We present an alternative approach to developing systems to serve the needs of scientific computing

- Choose our science target first to drive design decisions
- Leverage new technologies driven by consumer market
- ***Auto-tune software*** for performance, productivity, and portability
- Use hardware-accelerated architectural emulation to rapidly prototype designs (***auto-tune the hardware too!***)

- **Requires a holistic approach: Must innovate algorithm/software/hardware together (Co-tuning)**

Achieve 100x energy efficiency improvement over mainstream HPC approach

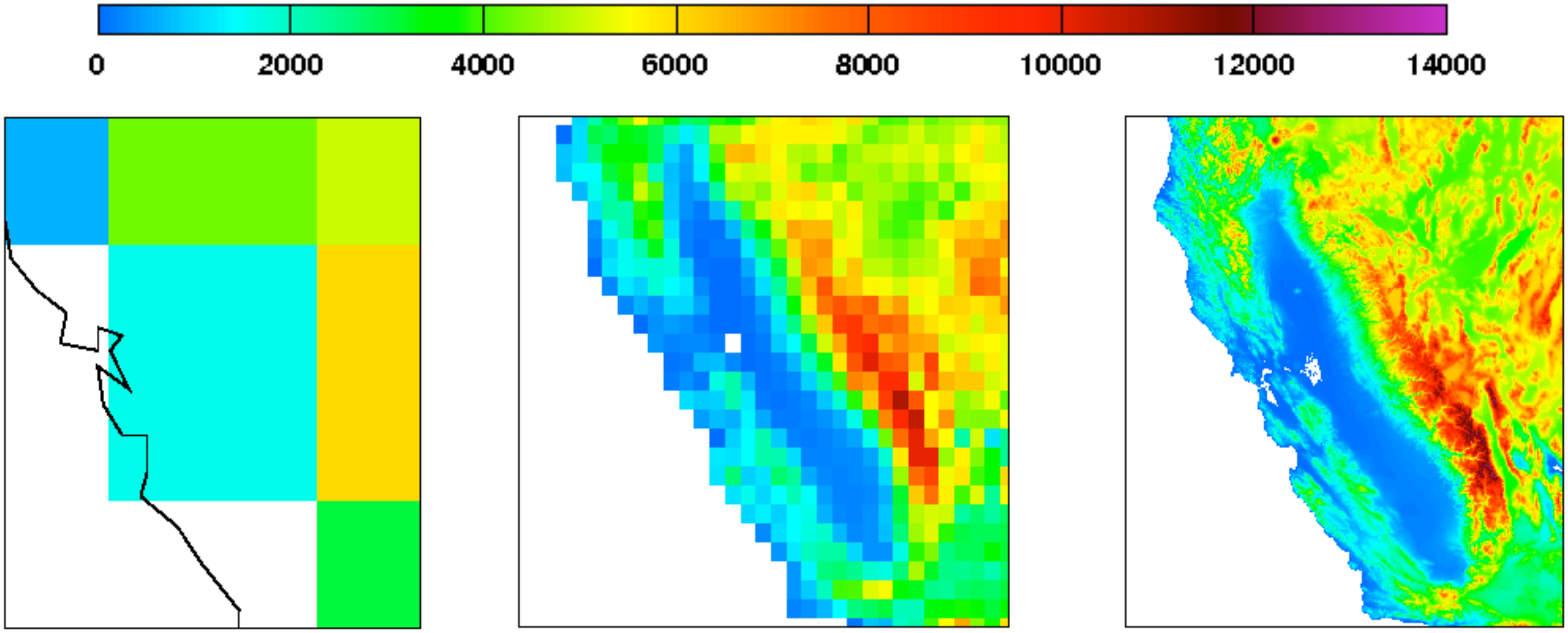
An Application Driver:
Global Cloud Resolving Climate Model

NERSC
NATIONAL ENERGY RESEARCH
SCIENTIFIC COMPUTING CENTER

Identify Target First!

(Global Cloud Resolving Climate Model)

Surface Altitude (feet)

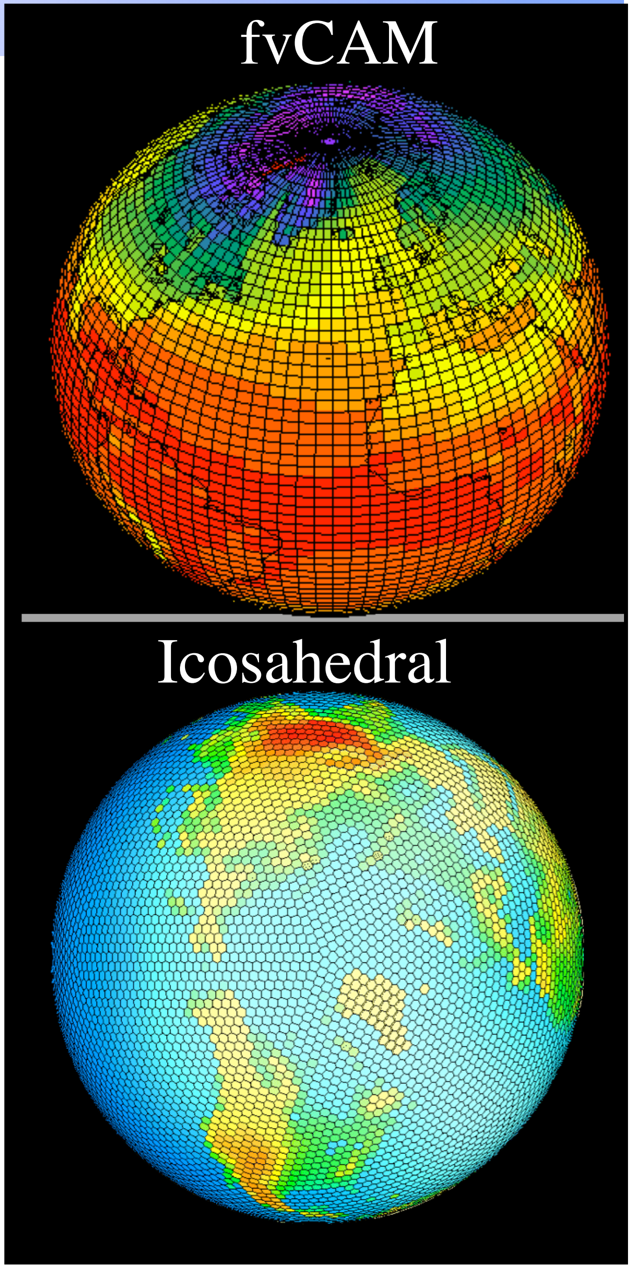


200km	25km	1km
Typical resolution of IPCC AR4 models	Upper limit of climate models with cloud parameterizations	Cloud system resolving models are a transformational change

Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

Computational Requirements for 1km Climate Model



Must maintain 1000x faster than real time for practical climate simulation

- **~2 million horizontal subdomains**
- **100 Terabytes of Memory**
  - **5MB memory per subdomain**
- **~20 million total subdomains**
  - **20 PF sustained *(200PF peak)***
  - **Nearest-neighbor communication**

- ***New discretization for climate model***
  - ***CSU Icosahedral Code***

Energy Efficient Hardware Building Blocks

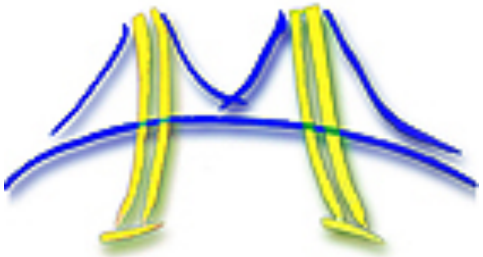
Mark Horowitz 2007: *"Years of research in low-power embedded computing have shown only one design technique to reduce power:* ***reduce waste.****"*

Seymour Cray 1977: *"Don't put anything in to a supercomputer that isn't necessary."*

Hardware: What are the problems?

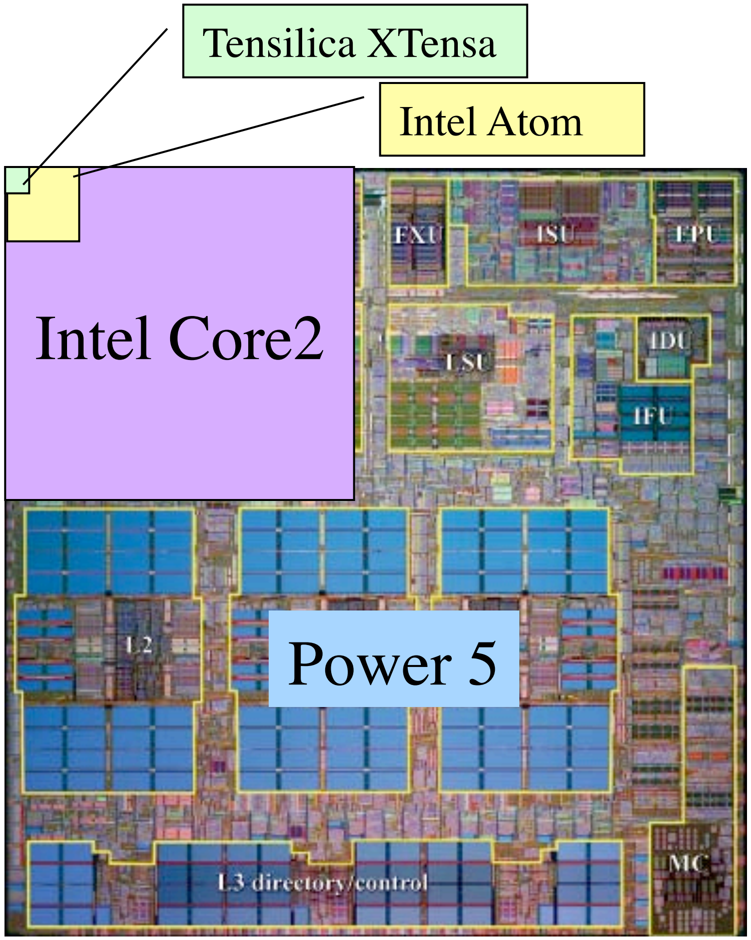
(Lessons from the Berkeley View)

- **Current Hardware/Lithography Constraints**
  - **Power limits leading edge chip designs**
    - Intel Tejas Pentium 4 cancelled due to power issues
  - **Yield on leading edge processes dropping dramatically**
    - IBM quotes yields of 10 – 20% on 8-processor Cell
  - **Design/validation leading edge chip is becoming unmanageable**
    - Verification teams > design teams on leading edge processors



- **Solution: Small Is Beautiful**
  - **Simpler (5- to 9-stage pipelined) CPU cores**
    - Small cores not much slower than large cores
  - **Parallel is energy efficient path to performance:CV^2F**
    - Lower threshold and supply voltages lowers energy per op
  - **Redundant processors can improve chip yield**
    - Cisco Metro 188 CPUs + 4 spares; Sun Niagara sells 6 or 8 CPUs
  - **Small, regular processing elements easier to verify**

Low-Power Design Principles



- **Cubic power improvement with lower clock rate due to V^2F**

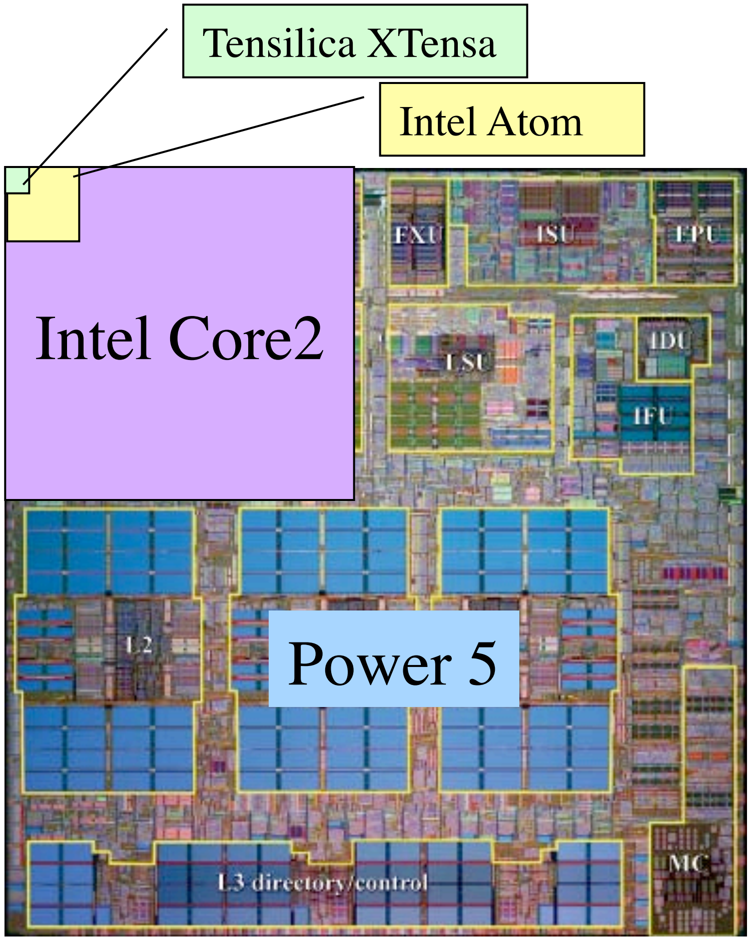
- **Slower clock rates enable use of simpler cores**

- **Simpler cores use less area (lower leakage) and reduce cost**

- **Tailor design to application to REDUCE WASTE**

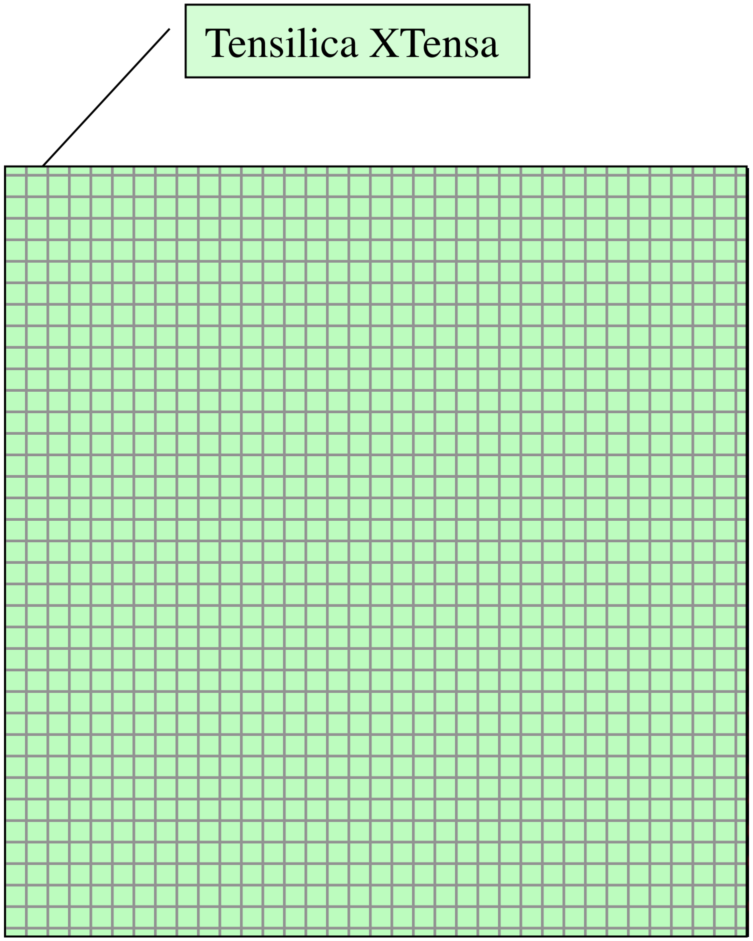
This is how iPhones and MP3 players are designed to maximize battery life and minimize cost

Low-Power Design Principles



- Power5 (server)
  - 120W@1900MHz
  - Baseline
- Intel Core2 sc (laptop) :
  - 15W@1000MHz
  - *4x more FLOPs/watt than baseline*
- Intel Atom (handhelds)
  - 0.625W@800MHz
  - 80x more
- Tensilica XTensa DP (Moto Razor) :
  - 0.09W@600MHz
  - 400x more *(80x-120x sustained)*

Low Power Design Principles

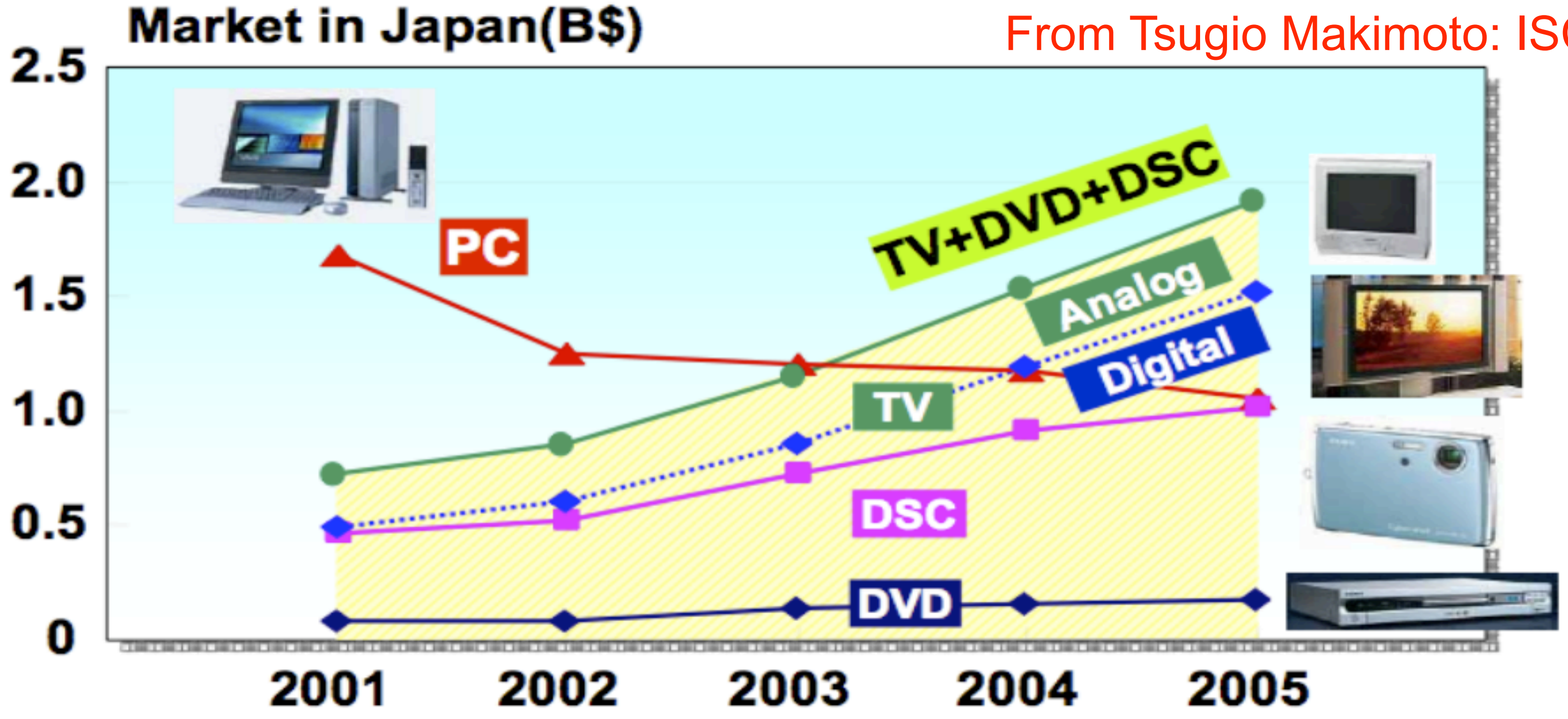


- Power5 (server)
  - 120W@1900MHz
  - Baseline
- Intel Core2 sc (laptop) :
  - 15W@1000MHz
  - *4x more FLOPs/watt than baseline*
- Intel Atom (handhelds)
  - 0.625W@800MHz
  - 80x more
- Tensilica XTensa DP (Moto Razor) :
  - 0.09W@600MHz
  - 400x more (80x-100x sustained)

Even if each simple core is 1/4th as computationally efficient as complex core, you can fit hundreds of them on a single chip and still be 100x more power efficient.

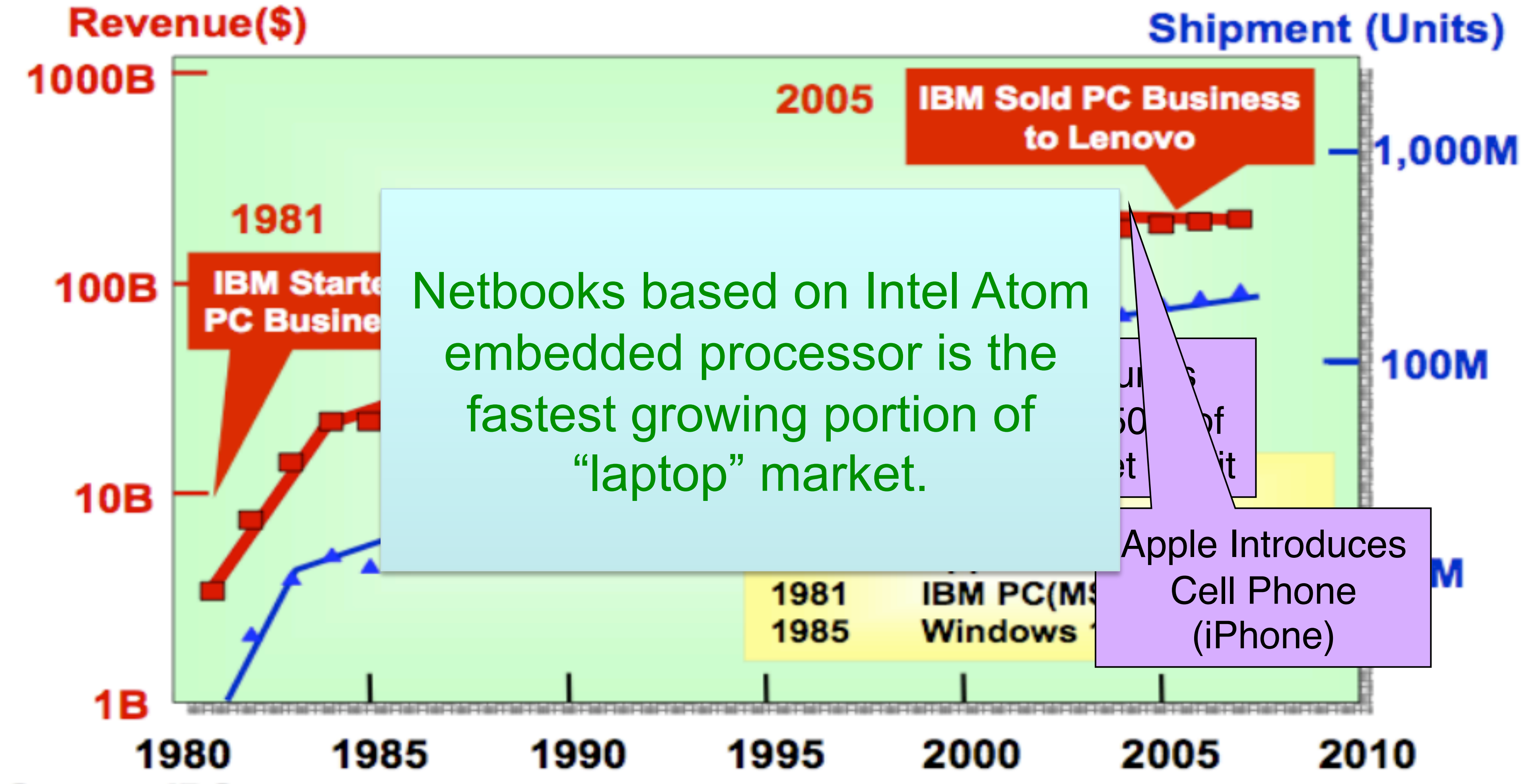
Technology Investment Trends

- 1990s - R&D computing hardware dominated by desktop/COTS
  - Had to learn how to use COTS technology for HPC
- 2010 - R&D investments moving rapidly to consumer electronics/ embedded processing
  - Must learn how to leverage embedded processor technology for future HPC systems



From Tsugio Makimoto: ISC2006

Consumer Electronics has Replaced PCs as the Dominant Market Force in CPU Design!!



Netbooks based on Intel Atom embedded processor is the fastest growing portion of “laptop” market.

Apple Introduces Cell Phone (iPhone)

Source: IDC

From Tsugio Makimoto: ISC2006

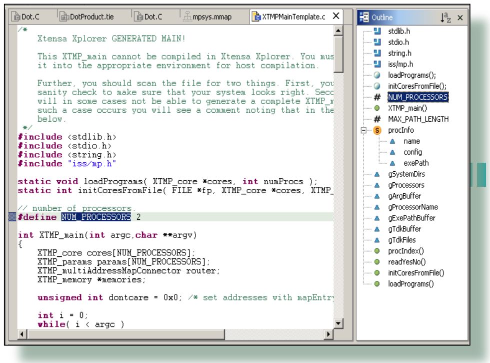
Embracing the Embedded Market

- **Have most of the IP and experience with for low-power technology**
- **Have sophisticated tools for rapid turn-around of designs**
- **Vibrant commodity market in IP components**
  - ***Change your notion of "commodity"!***
  - ***it's commodity IP on the chip (not the chip itself!)***

- **Convergence with HPC requirements**
  - Need better computational efficiency and lower power
  - Now we both must face parallelism

Embedded Design Automation

(Example from Existing Tensilica Design Flow)

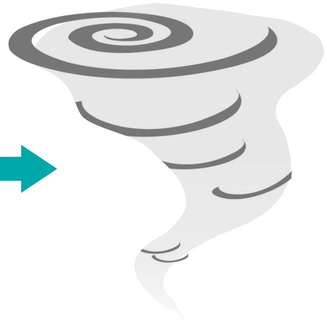


```
/*
    Xtensa Xplorer GENERATED MAIN!

    This XTMP_main cannot be compiled in Xtensa Xplorer. You mus
    it into the appropriate environment for host compilation.

    Further, you should scan the file for two things. First, you
    sanity check to make sure that your system looks right. Seco
    will in some cases not be able to generate a complete XTMP_m
    such a case occurs you will see a comment noting that in the
    below.
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "iss/mp.h"

static void loadPrograms( XTMP_core *cores, int numProcs );
static int initCoresFromFile( FILE *fp, XTMP_core *cores, XTMP_

// number of processors.
#define NUM_PROCESSORS 2

int XTMP_main(int argc,char **argv)
{
    XTMP_core cores[NUM_PROCESSORS];
    XTMP_params params[NUM_PROCESSORS];
    XTMP_multiAddressMapConnector router;
    XTMP_memory *memories;

    unsigned int dontcare = 0x0; /* set addresses with mapEntry

    int i = 0;
    while( i < argc )
```

```
stdlib.h
stdio.h
string.h
iss/mp.h
loadPrograms();
initCoresFromFile();
NUM_PROCESSORS
XTMP_main()
MAX_PATH_LENGTH
procInfo
    name
    config
    exePath
gSystemDirs
gProcessors
gArgBuffer
gProcessorName
gExePathBuffer
gTdkBuffer
gTdkFiles
procIndex()
readYesNo()
initCoresFromFile()
loadPrograms()
```

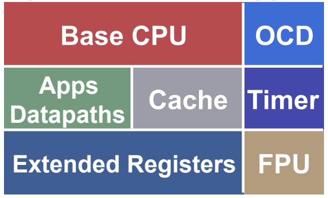
Processor configuration

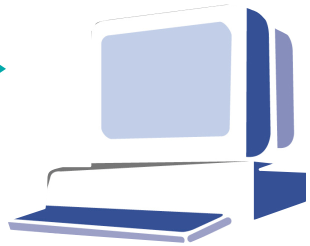
1. Select from menu
2. Automatic instruction discovery (XPRES Compiler)
3. Explicit instruction description (TIE)



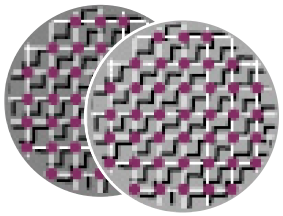
Processor Generator (*Tensilica*)

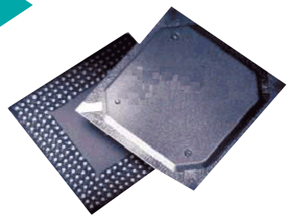
Application-optimized processor implementation (RTL/Verilog)





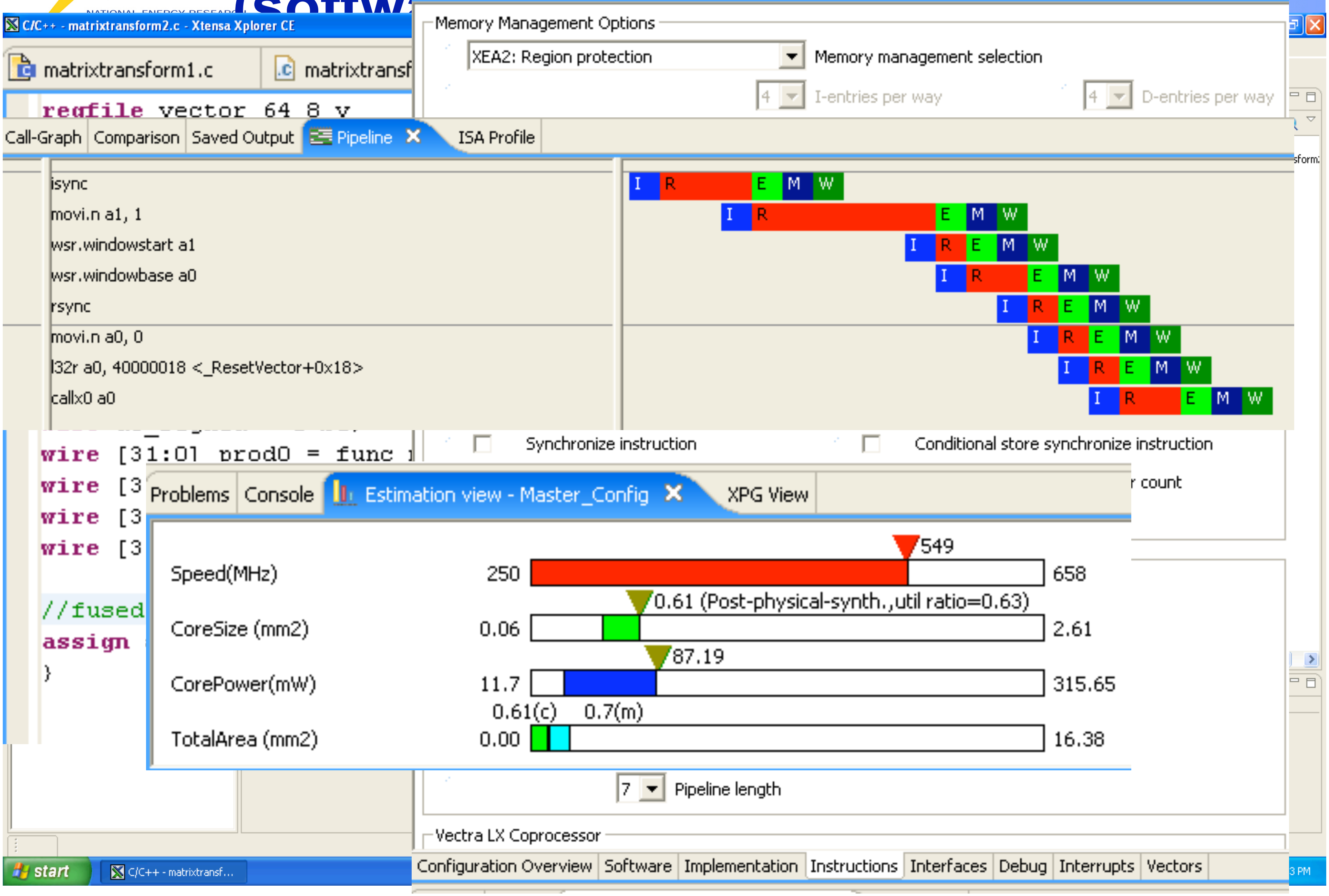
Tailored SW Tools: Compiler, debugger, simulators, Linux, other OS Ports *(Automatically generated together with the Core)*





Build with any process in any fab

Processor Generator
(software modeling for triage)



C/C++ - matrixtransform2.c - Xtensa Xplorer CE

matrixtransform1.c | matrixtransf

```
regfile vector 64 8 v
wire [31:0] prod0 = func
wire [3
wire [3
wire [3

//fused
assign
}
```

Memory Management Options

XEA2: Region protection — Memory management selection

4 I-entries per way — 4 D-entries per way

Call-Graph | Comparison | Saved Output | Pipeline | ISA Profile

```
isync                                    I R E M W
movi.n a1, 1                             I R E M W
wsr.windowstart a1                       I R E M W
wsr.windowbase a0                        I R E M W
rsync                                    I R E M W
movi.n a0, 0                             I R E M W
l32r a0, 40000018 <_ResetVector+0x18>    I R E M W
callx0 a0                                I R E M W
```

□ Synchronize instruction — □ Conditional store synchronize instruction

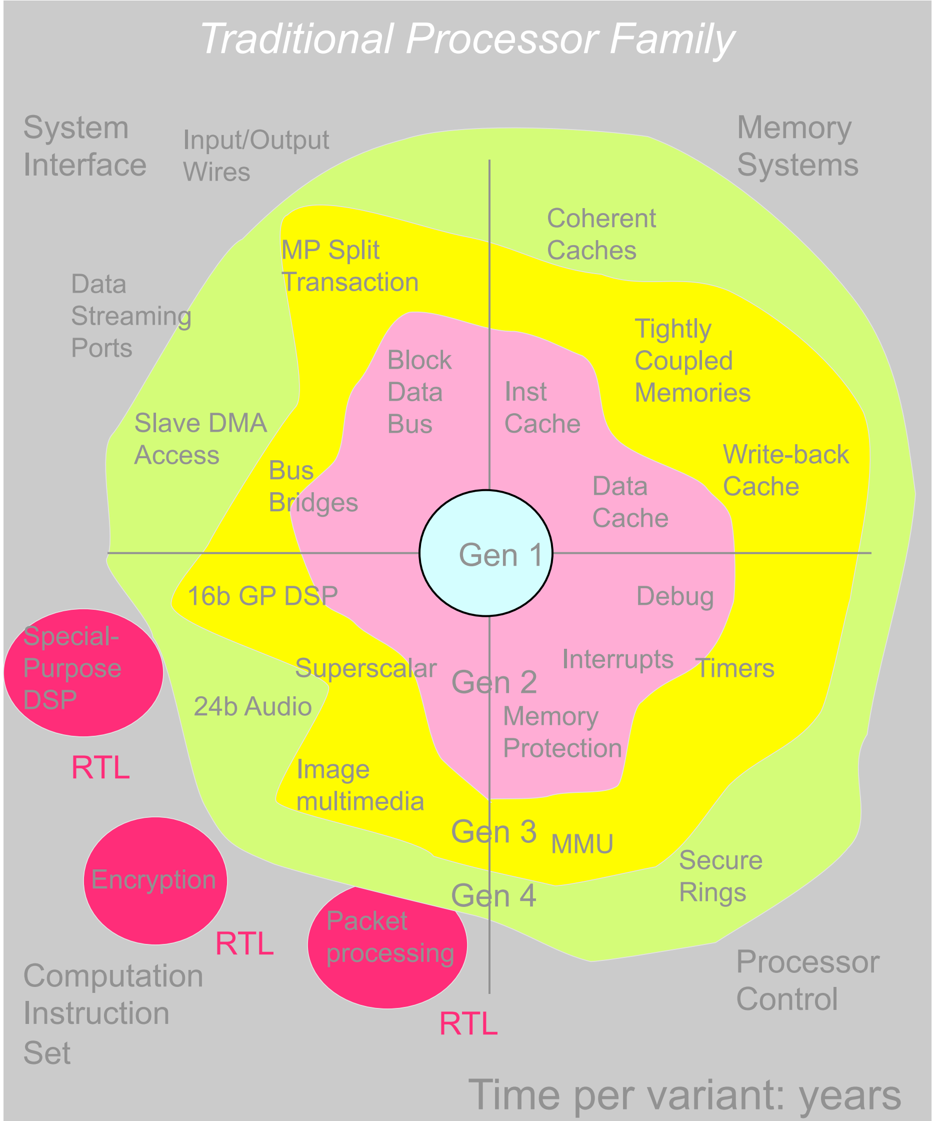
Problems | Console | Estimation view - Master_Config | XPG View

	Min	Value	Max
Speed(MHz)	250	549	658
CoreSize (mm2)	0.06	0.61 (Post-physical-synth.,util ratio=0.63)	2.61
CorePower(mW)	11.7	87.19	315.65
TotalArea (mm2)	0.00	0.61(c) 0.7(m)	16.38

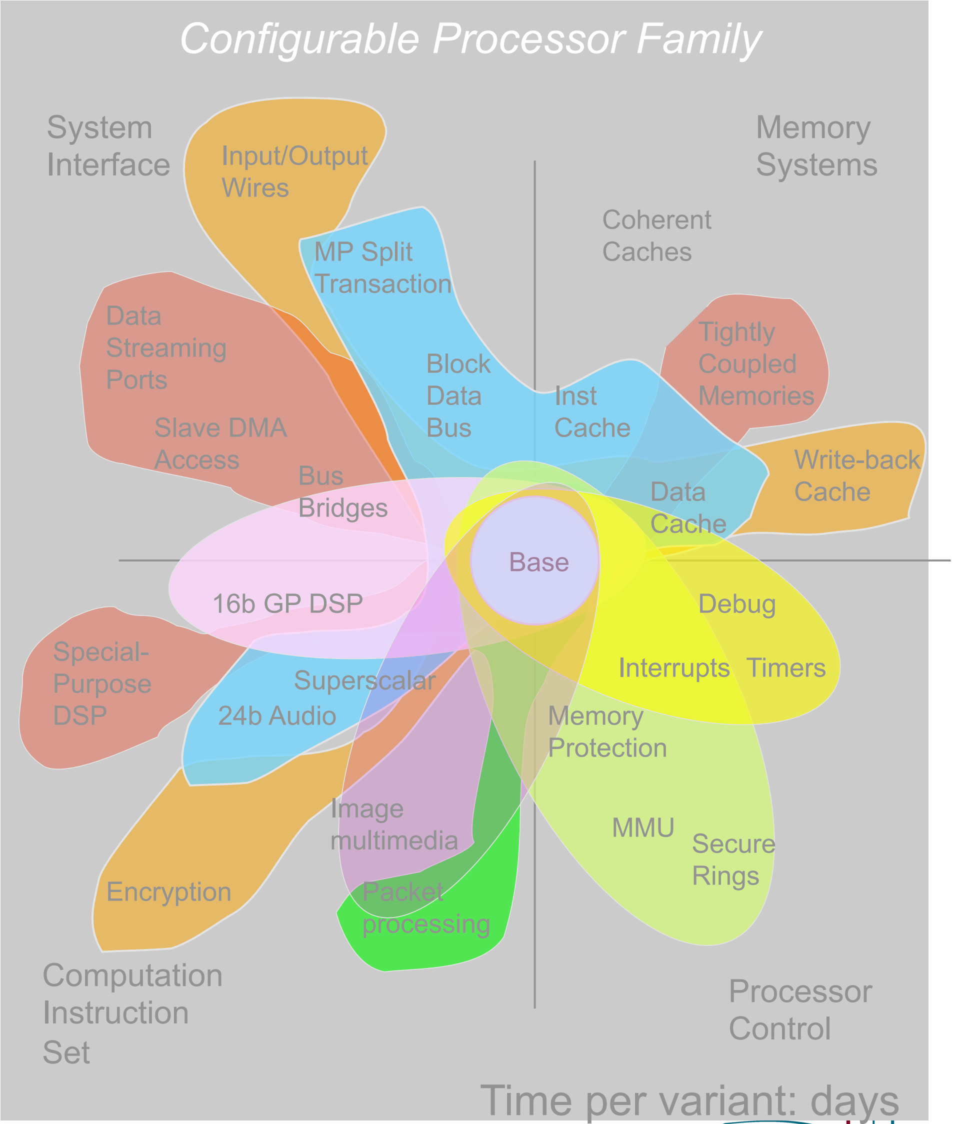
7 Pipeline length

Vectra LX Coprocessor

Peel Back the Historical Growth of Instruction Sets *(accretion of junk!)*


Traditional Processor Family

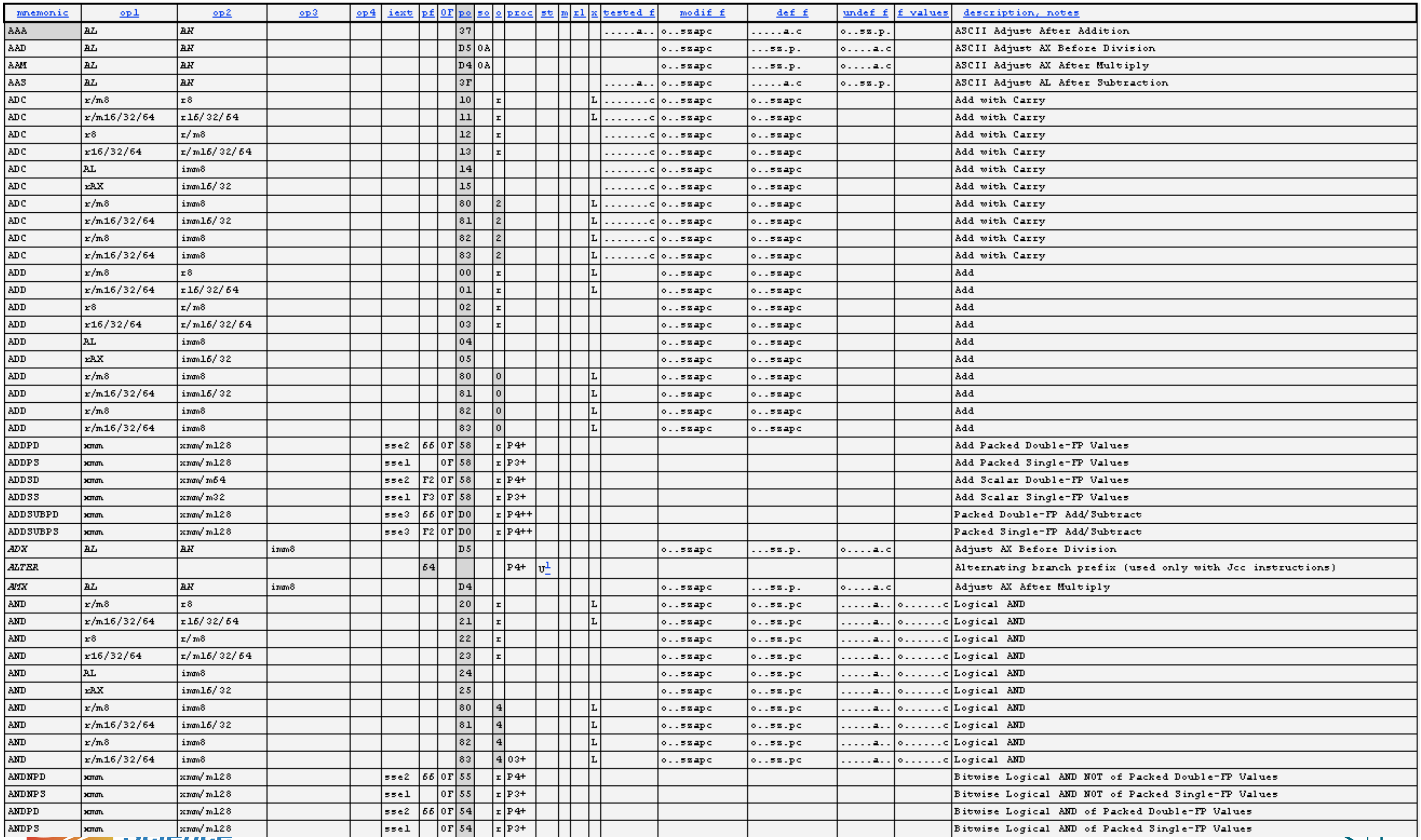
Time per variant: years


Configurable Processor Family

Time per variant: days

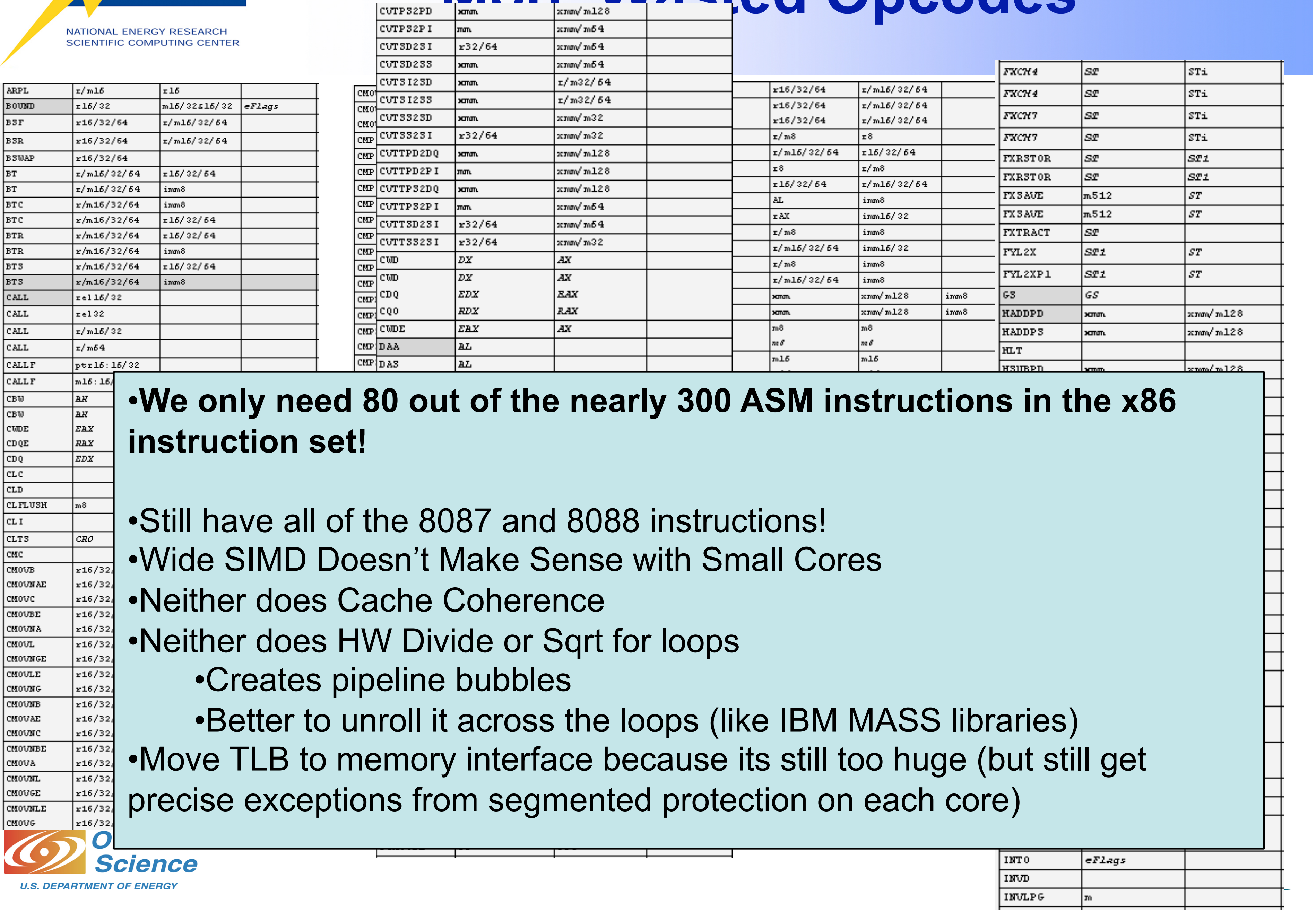
⬭ Area = silicon cost and power

A Short List of x86 Opcodes that Science Applications Don’t Need!



mnemonic	op1	op2	op3	op4	iext	pf	0F	po	so	o	proc	st	m	rl	x	tested f	modif f	def f	undef f	f values	description, notes
AAA	AL	AH						37								.....a..	o..szapc	.....a.c	o..sz.p.		ASCII Adjust After Addition
AAD	AL	AH						D5	0A								o..szapc	...sz.p.	o....a.c		ASCII Adjust AX Before Division
AAM	AL	AH						D4	0A								o..szapc	...sz.p.	o....a.c		ASCII Adjust AX After Multiply
AAS	AL	AH						3F								.....a..	o..szapc	.....a.c	o..sz.p.		ASCII Adjust AL After Subtraction
ADC	r/m8	r8						10		r					L	c	o..szapc	o..szapc			Add with Carry
ADC	r/m16/32/64	r16/32/64						11		r					L	c	o..szapc	o..szapc			Add with Carry
ADC	r8	r/m8						12		r						c	o..szapc	o..szapc			Add with Carry
ADC	r16/32/64	r/m16/32/64						13		r						c	o..szapc	o..szapc			Add with Carry
ADC	AL	imm8						14								c	o..szapc	o..szapc			Add with Carry
ADC	rAX	imm16/32						15								c	o..szapc	o..szapc			Add with Carry
ADC	r/m8	imm8						80		2					L	c	o..szapc	o..szapc			Add with Carry
ADC	r/m16/32/64	imm16/32						81		2					L	c	o..szapc	o..szapc			Add with Carry
ADC	r/m8	imm8						82		2					L	c	o..szapc	o..szapc			Add with Carry
ADC	r/m16/32/64	imm8						83		2					L	c	o..szapc	o..szapc			Add with Carry
ADD	r/m8	r8						00		r					L		o..szapc	o..szapc			Add
ADD	r/m16/32/64	r16/32/64						01		r					L		o..szapc	o..szapc			Add
ADD	r8	r/m8						02		r							o..szapc	o..szapc			Add
ADD	r16/32/64	r/m16/32/64						03		r							o..szapc	o..szapc			Add
ADD	AL	imm8						04									o..szapc	o..szapc			Add
ADD	rAX	imm16/32						05									o..szapc	o..szapc			Add
ADD	r/m8	imm8						80		0					L		o..szapc	o..szapc			Add
ADD	r/m16/32/64	imm16/32						81		0					L		o..szapc	o..szapc			Add
ADD	r/m8	imm8						82		0					L		o..szapc	o..szapc			Add
ADD	r/m16/32/64	imm8						83		0					L		o..szapc	o..szapc			Add
ADDPD	xmm	xmm/m128			sse2	66	0F	58		r	P4+										Add Packed Double-FP Values
ADDPS	xmm	xmm/m128			sse1		0F	58		r	P3+										Add Packed Single-FP Values
ADDSD	xmm	xmm/m64			sse2	F2	0F	58		r	P4+										Add Scalar Double-FP Values
ADDSS	xmm	xmm/m32			sse1	F3	0F	58		r	P3+										Add Scalar Single-FP Values
ADDSUBPD	xmm	xmm/m128			sse3	66	0F	D0		r	P4++										Packed Double-FP Add/Subtract
ADDSUBPS	xmm	xmm/m128			sse3	F2	0F	D0		r	P4++										Packed Single-FP Add/Subtract
ADX	AL	AH	imm8					D5									o..szapc	...sz.p.	o....a.c		Adjust AX Before Division
ALTER						64					P4+	U[1]									Alternating branch prefix (used only with Jcc instructions)
AMX	AL	AH	imm8					D4									o..szapc	...sz.p.	o....a.c		Adjust AX After Multiply
AND	r/m8	r8						20		r					L		o..szapc	o..sz.pc	.....a..	o......c	Logical AND
AND	r/m16/32/64	r16/32/64						21		r					L		o..szapc	o..sz.pc	.....a..	o......c	Logical AND
AND	r8	r/m8						22		r							o..szapc	o..sz.pc	.....a..	o......c	Logical AND
AND	r16/32/64	r/m16/32/64						23		r							o..szapc	o..sz.pc	.....a..	o......c	Logical AND
AND	AL	imm8						24									o..szapc	o..sz.pc	.....a..	o......c	Logical AND
AND	rAX	imm16/32						25									o..szapc	o..sz.pc	.....a..	o......c	Logical AND
AND	r/m8	imm8						80		4					L		o..szapc	o..sz.pc	.....a..	o......c	Logical AND
AND	r/m16/32/64	imm16/32						81		4					L		o..szapc	o..sz.pc	.....a..	o......c	Logical AND
AND	r/m8	imm8						82		4					L		o..szapc	o..sz.pc	.....a..	o......c	Logical AND
AND	r/m16/32/64	imm8						83		4	03+				L		o..szapc	o..sz.pc	.....a..	o......c	Logical AND
ANDNPD	xmm	xmm/m128			sse2	66	0F	55		r	P4+										Bitwise Logical AND NOT of Packed Double-FP Values
ANDNPS	xmm	xmm/m128			sse1		0F	55		r	P3+										Bitwise Logical AND NOT of Packed Single-FP Values
ANDPD	xmm	xmm/m128			sse2	66	0F	54		r	P4+										Bitwise Logical AND of Packed Double-FP Values
ANDPS	xmm	xmm/m128			sse1		0F	54		r	P3+										Bitwise Logical AND of Packed Single-FP Values

More Wasted Opcodes



ARPL	r/m16	r16	
BOUND	r16/32	m16/32&16/32	eFlags
BSF	r16/32/64	r/m16/32/64	
BSR	r16/32/64	r/m16/32/64	
BSWAP	r16/32/64		
BT	r/m16/32/64	r16/32/64	
BT	r/m16/32/64	imm8	
BTC	r/m16/32/64	imm8	
BTC	r/m16/32/64	r16/32/64	
BTR	r/m16/32/64	r16/32/64	
BTR	r/m16/32/64	imm8	
BTS	r/m16/32/64	r16/32/64	
BTS	r/m16/32/64	imm8	
CALL	rel16/32		
CALL	rel32		
CALL	r/m16/32		
CALL	r/m64		
CALLF	ptr16:16/32		
CALLF	m16:16/		
CBW	AH		
CBW CWDE CDQE	AH EAX RAX		
CDQ	EDX		
CLC			
CLD			
CLFLUSH	m8		
CLI			
CLTS	CR0		
CMC			
CMOVB CMOVNAE CMOVC	r16/32, r16/32, r16/32,		
CMOVBE CMOVNA	r16/32, r16/32,		
CMOVL CMOVNGE	r16/32, r16/32,		
CMOVLE CMOVNG	r16/32, r16/32,		
CMOVNB CMOVAE CMOVNC	r16/32, r16/32, r16/32,		
CMOVNBE CMOVA	r16/32, r16/32,		
CMOVNL CMOVGE	r16/32, r16/32,		
CMOVNLE CMOVG	r16/32, r16/32,		

CMO
CMO
CMO
CMP
CMP
CMP
CMP
CMP
CMP
CMP
CMP
CMP
CMP
CMP
CMP
CMP
CMP
CMP

CVTPS2PD	xmm	xmm/m128	
CVTPS2PI	mm	xmm/m64	
CVTSD2SI	r32/64	xmm/m64	
CVTSD2SS	xmm	xmm/m64	
CVTSI2SD	xmm	r/m32/64	
CVTSI2SS	xmm	r/m32/64	
CVTSS2SD	xmm	xmm/m32	
CVTSS2SI	r32/64	xmm/m32	
CVTTPD2DQ	xmm	xmm/m128	
CVTTPD2PI	mm	xmm/m128	
CVTTPS2DQ	xmm	xmm/m128	
CVTTPS2PI	mm	xmm/m64	
CVTTSD2SI	r32/64	xmm/m64	
CVTTSS2SI	r32/64	xmm/m32	
CWD	DX	AX	
CWD CDQ CQO	DX EDX RDX	AX EAX RAX	
CWDE	EAX	AX	
DAA	AL		
DAS	AL		

	r16/32/64	r/m16/32/64	
	r16/32/64 r16/32/64	r/m16/32/64 r/m16/32/64	
	r/m8	r8	
	r/m16/32/64	r16/32/64	
	r8	r/m8	
	r16/32/64	r/m16/32/64	
	AL	imm8	
	rAX	imm16/32	
	r/m8	imm8	
	r/m16/32/64	imm16/32	
	r/m8	imm8	
	r/m16/32/64	imm8	
	xmm	xmm/m128	imm8
	xmm	xmm/m128	imm8
	m8 m8	m8 m8	
	m16	m16	

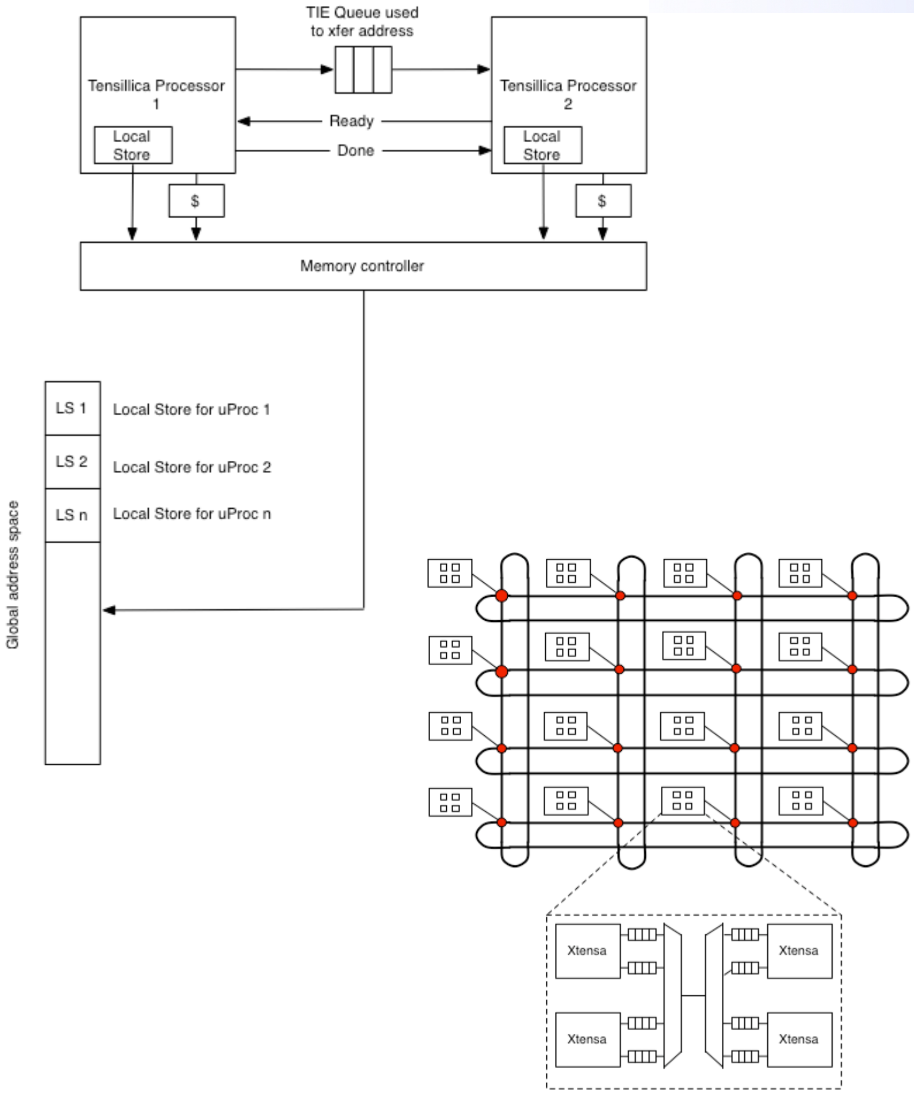
FXCH4	ST	STi
FXCH4	ST	STi
FXCH7	ST	STi
FXCH7	ST	STi
FXRSTOR	ST	ST1
FXRSTOR	ST	ST1
FXSAVE	m512	ST
FXSAVE	m512	ST
FXTRACT	ST	
FYL2X	ST1	ST
FYL2XP1	ST1	ST
GS	GS	
HADDPD	xmm	xmm/m128
HADDPS	xmm	xmm/m128
HLT		
HSUBPD	xmm	xmm/m128
INTO	eFlags	
INVD		
INVLPG	m	

- **We only need 80 out of the nearly 300 ASM instructions in the x86 instruction set!**

- Still have all of the 8087 and 8088 instructions!
- Wide SIMD Doesn't Make Sense with Small Cores
- Neither does Cache Coherence
- Neither does HW Divide or Sqrt for loops
  - Creates pipeline bubbles
  - Better to unroll it across the loops (like IBM MASS libraries)
- Move TLB to memory interface because its still too huge (but still get precise exceptions from segmented protection on each core)

BLDG AIR INTAKE

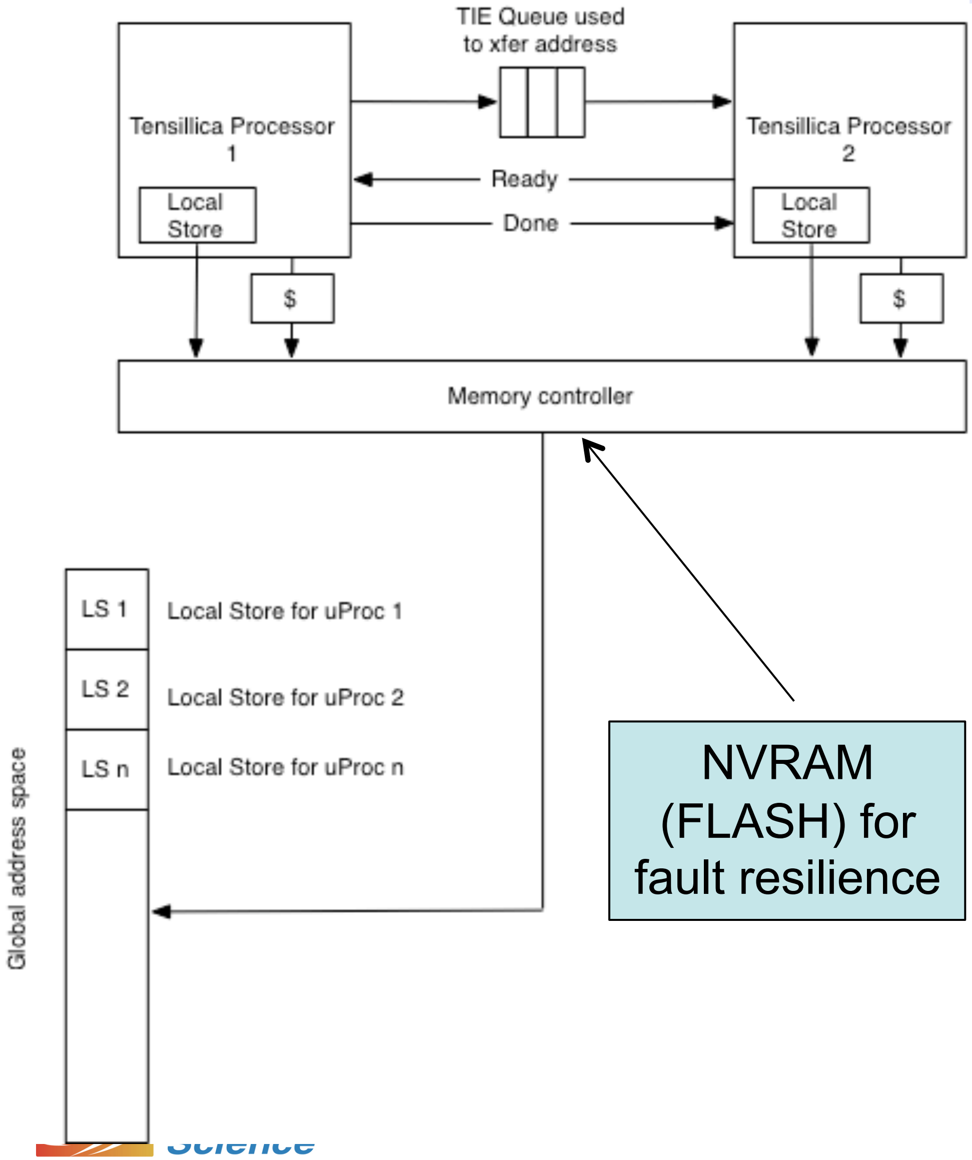
Science-Optimized Processor Design



	Intel Core2 (Penryn)	Intel Atom core	Tensilica core w/ 64-bit FP
Die area (mm^2)	53.5	25	5.32
Process	45 nm	45 nm	65 nm
Power	18W	0.625W	0.091W
Freq	2930 MHz	800MHz	500MHz
Flops / Watt	162	1280	***4065***

Architectural Support for PGAS

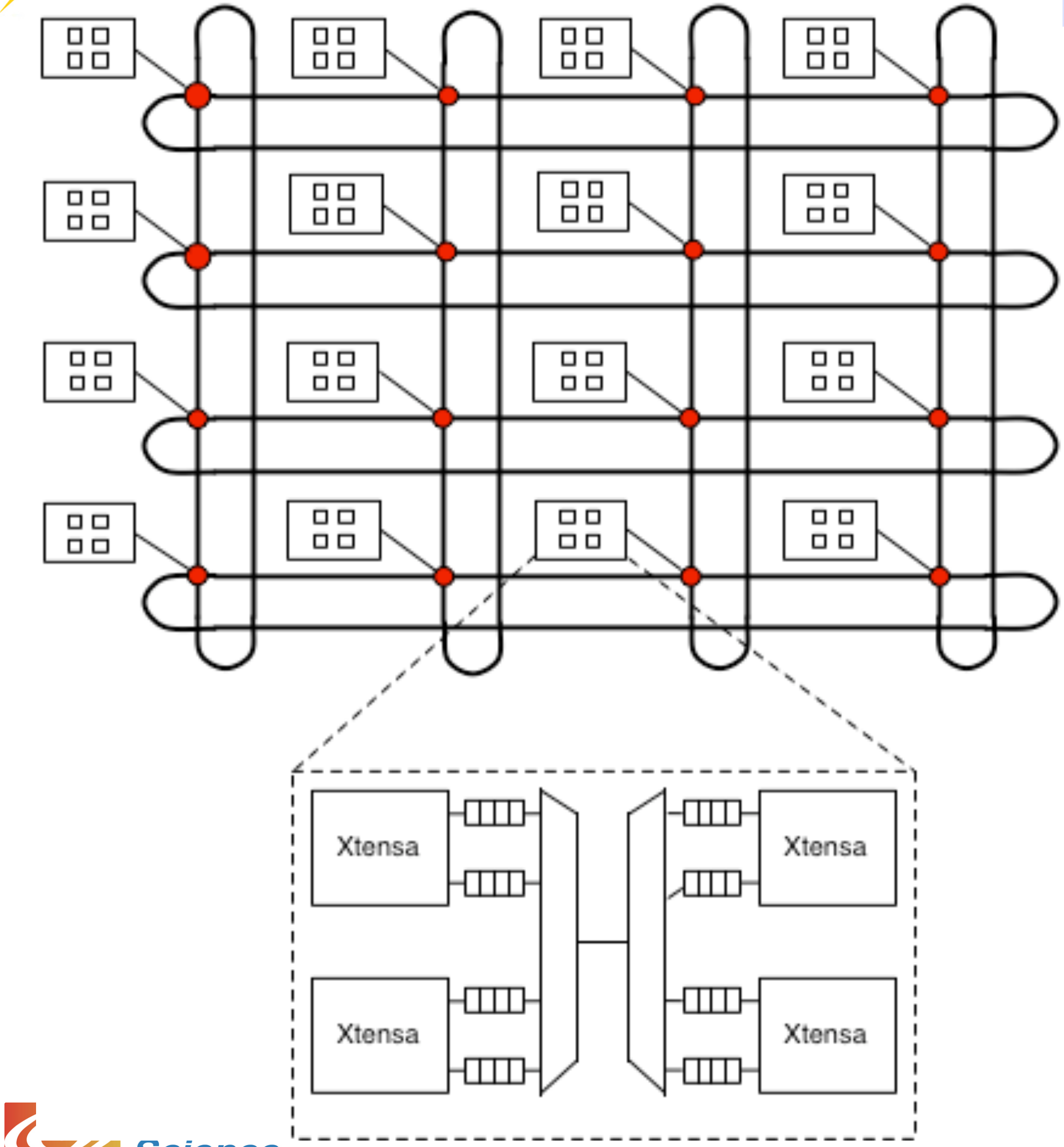
Make hardware easier to program!



NVRAM (FLASH) for fault resilience

- Logical topology is a full crossbar
- Each local store mapped to global address space
- To initiate a DMA transfer between processors:
  - Processors exchange starting addresses through TIE Queue interface
    - Optimized for small transfers
  - When ready, copy done directly from LS to LS
  - Copy will bypass cache hierarchy

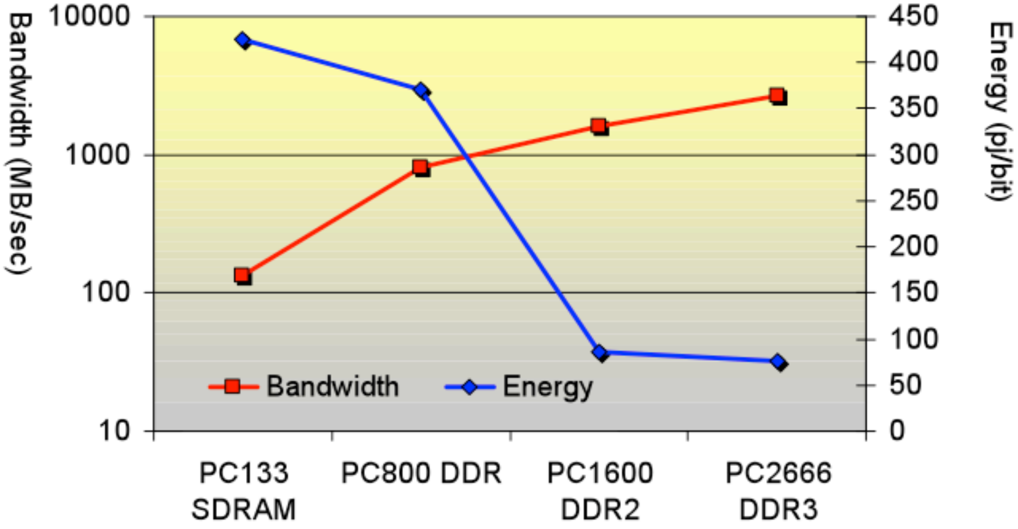
Network-on-Chip (NoC) Architecture

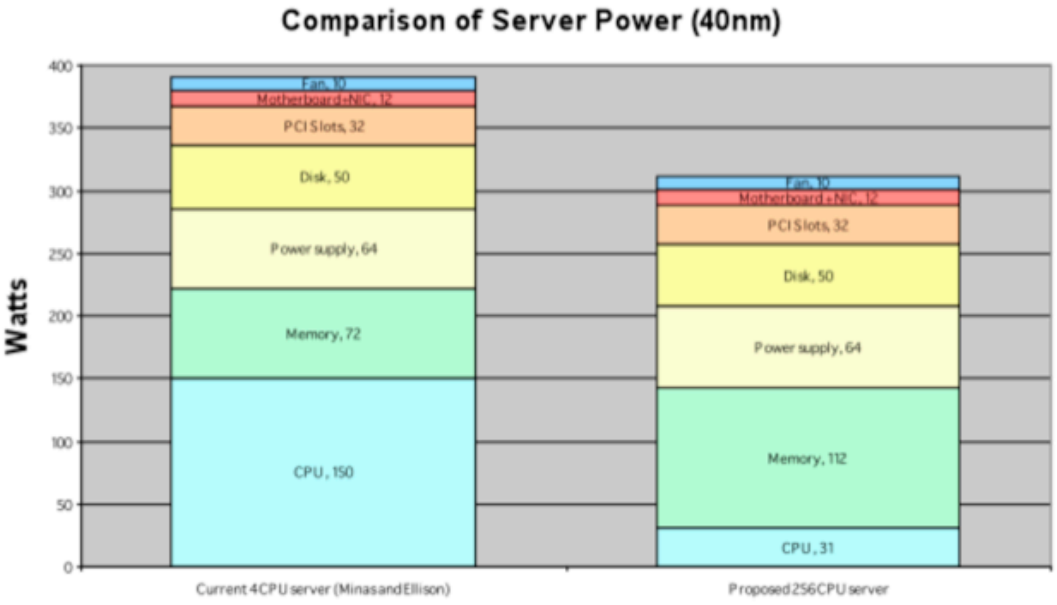


- Concentrated torus
  - Direct connect between 4 processors on a tile
  - Packet switched network connecting tiles
- Between 64 and 128 processors per die
- **Silicon Photonics as option for NoC**

What about Memory?

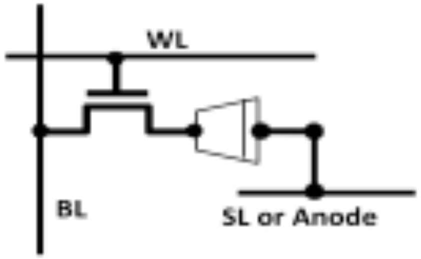
- Processor energy savings easily negated by high cost of DRAM power
- DRAM Power dominated by:
  - Sense amp
  - DDR Memory interface bus power
- Overfetch adds inefficiency to an already power hungry system





Looking Beyond DRAM

- Resistive Change RAM (ReRAM)
  - Nonvolatile - no refresh required!
  - No high-voltage requirement
  - Less energy / write (compared to FLASH)
  - More robust than FLASH
    - More cycles to cell wear out
  - Lower read energy than DRAM
    - < 1V read-out voltage
  - Similar density to flash
    - MLC capable
    - 2-4x DRAM
  - Read / write speeds comparable (or better!) than DRAM
  - Integrates very well with existing CMOS processes



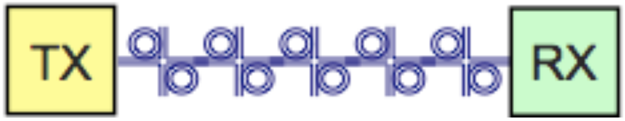
CATHODE
ANODE

Overall 10x reduction in power with a 4x increase in density

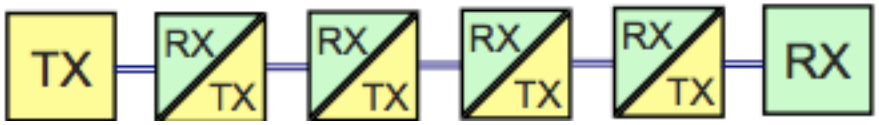
The problem with Wires: *Energy to move data proportional to distance*

- Wire cost to move a bit: (Telegraph Eqn.)
  - energy = bitrate * Length^2 / cross-section area
  - On-Chip (1cm): ~1pJ/bit, 100Tb/s
  - On-Module (5cm): ~2-5pJ/bit, 10Tb/s
  - On-Board (20cm): ~10pJ/bit, 1Tb/s
  - Intra-rack (1m): ~10-15pJ/bit, 1Tb/s
  - Inter-cabinet(2-50m): 15-30pJ/bit, 5-10Tb/s aggregate
- To move a bit with optics: target ~1-2pJ/bit for all distance scales

Photonics requires no redrive and passive switch little power



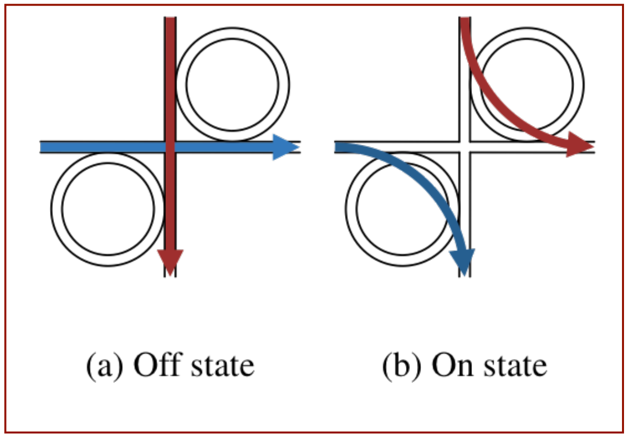
Copper requires to signal amplification even for on-chip connections

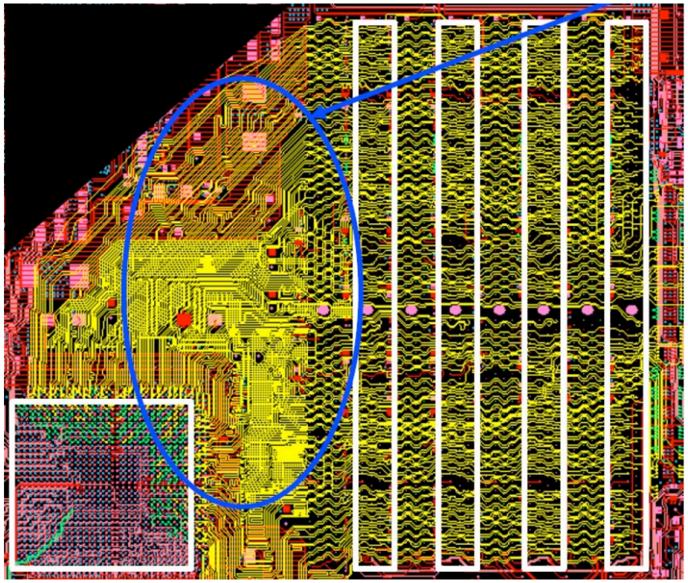


NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

Optical Interconnect

- On chip:
  - Optical interconnect enabled with Si photonic ring resonators
  - Integrates with conventional CMOS
  - Up to 27x power improvement
- Off Chip:
  - DDR interface power hungry
    - Cu line capacitance
    - Large voltage swing
  - Optical link much more efficient
    - Very small voltage modulation required
    - 50x reduction in interface power
- Unified optical fabric to reduce optical / electrical conversion
- Collaborating with Keren Bergmen's group at Columbia





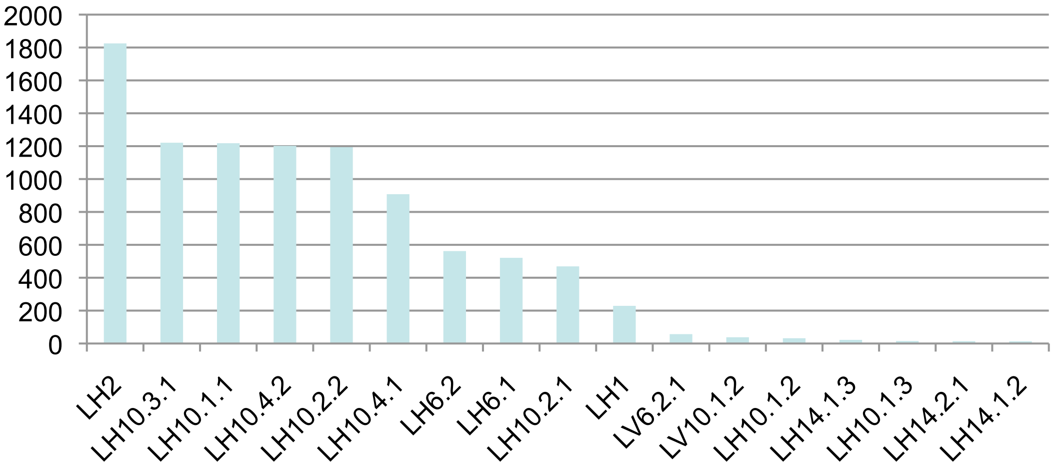
Wiring of a single channel DDR to the Memory controller (Intel)

Analyze Climate Code Memory Movement

Optimized Data Movement: Huge Savings in Energy Efficiency and Cost

- **Analyzed Each Loop of Climate code Individually**
- **Trace analysis key to memory requirements**
  - **Actually running the code gives realistic values for memory footprint, temporal reuse, DRAM bandwidth requirements**

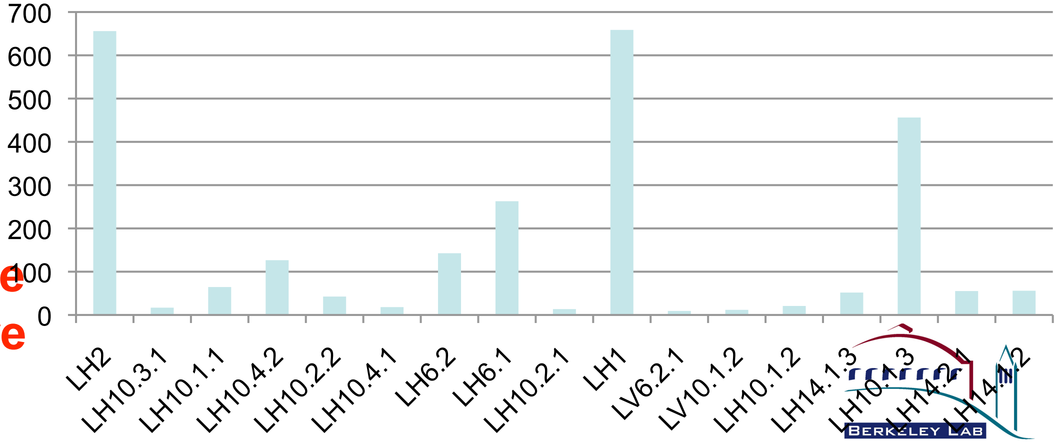
Memory footprint (KB)



- **Measure DRAM bandwidth for each loop!**
  - **(instruction throughput) X (memory footprint)/ (instruction counts)**

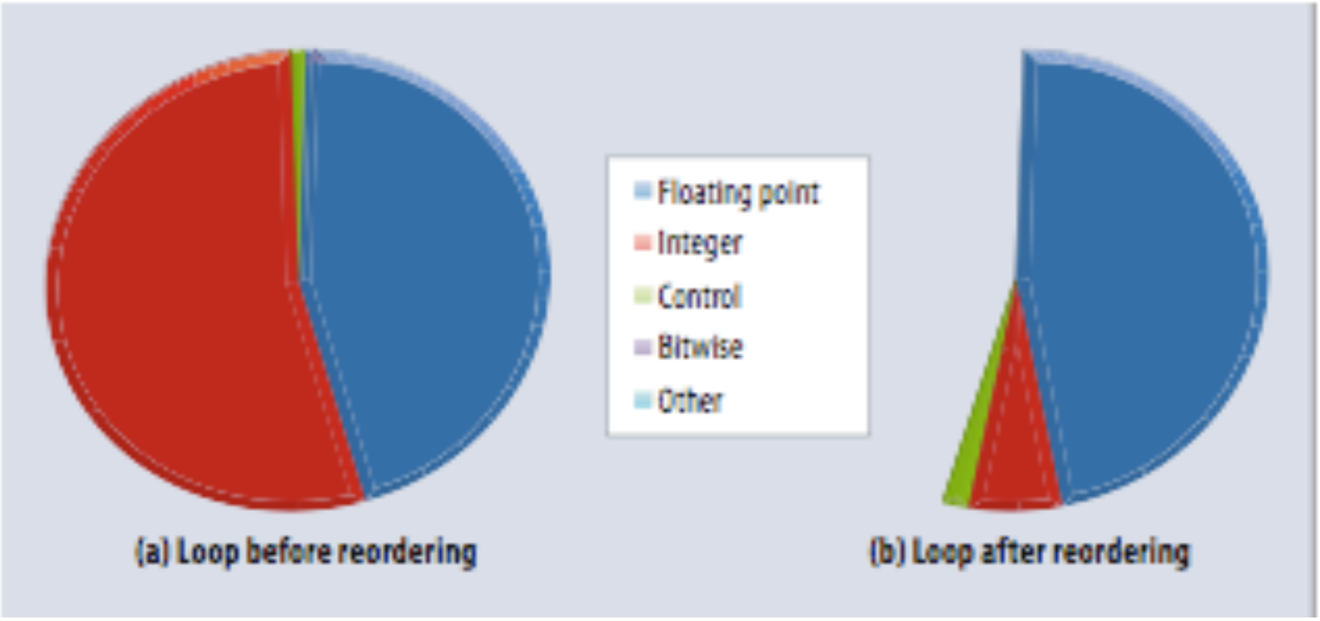
1-byte-per-FLOP could be reduced with local-store

Bandwidth Requirements (MB/s) (Instructions/Cycle=1, 500 MHz)



Optimizing Instruction Mix

LH2 (small domain) LH2 (small domain, reordered)



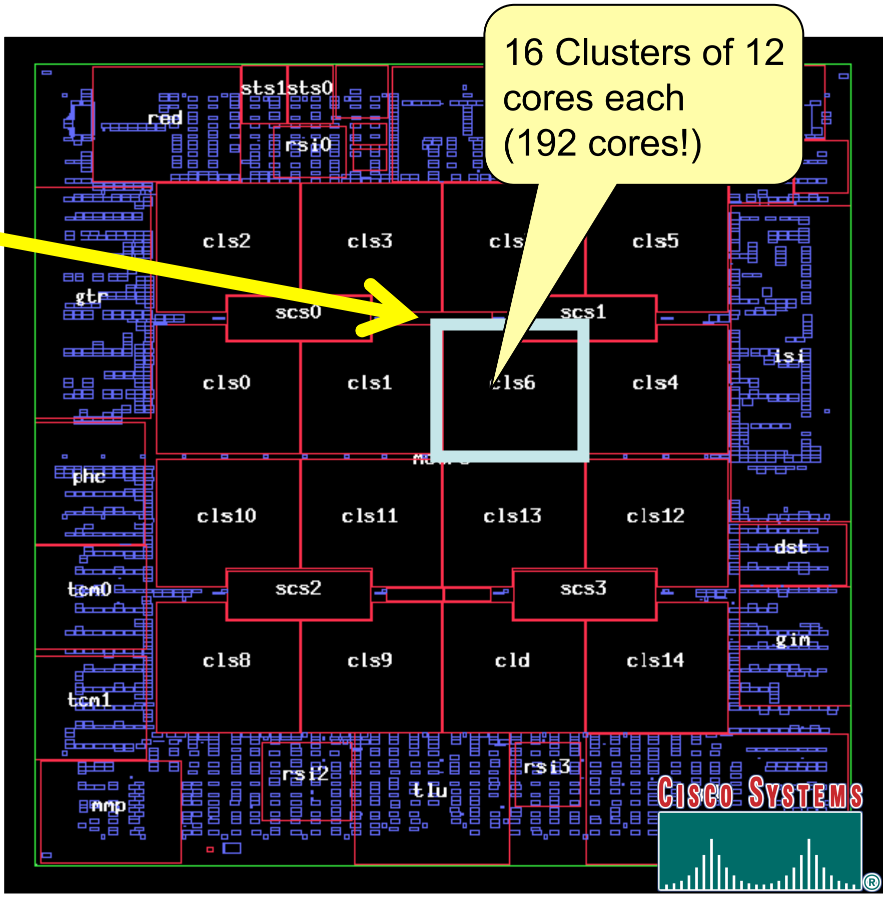
- Memory footprint: 160 KB
- Cache size requirement: 160 KB
- < 50% instructions are floating-point
    - Huge overhead for address generation
- Although code streams through data, loop ordering was bad ➔ cachelines reused although addresses were not

- Memory footprint: 160 KB
- Cache size requirement: 1 KB
- > 85% instructions are floating-point
    - Good ordering ➔ simpler addressing

160x reduction in cache size!
2x savings in execution time

Green Flash: Fault Tolerance/Resilience

- ***Large scale applications must tolerate node failures***
- **Our design does not expose unique risks**
  - Faults proportional to sockets (not cores) & silicon surface area
  - Low-power manycore uses less surface area and fewer sockets
- **Hard Errors**
  - Spare cores in design (Cisco Metro: 188 cores + 8 spares)
  - SystemOnChip design (fewer components→fewer sockets)
- **Soft Errors**
  - ECC for memory and caches
  - On-board NVRAM controller for localized checkpoint



16 Clusters of 12 cores each (192 cores!)

Software Performance

Software Auto-tuning: Don’t depend on a human to do a machine’s job.

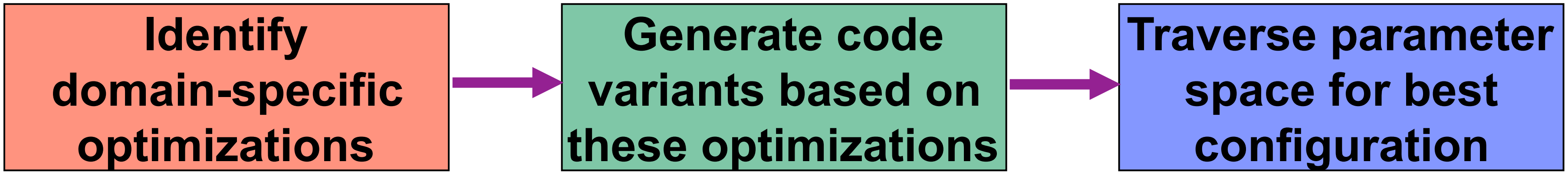
Auto-Tuning for Performance Portability

Challenge: How to optimize for multiple architectures

- Labor-intensive user optimizations for each specific architecture
- Different architectural solutions require vastly different optimizations
- Non-obvious interactions between optimizations & hardware

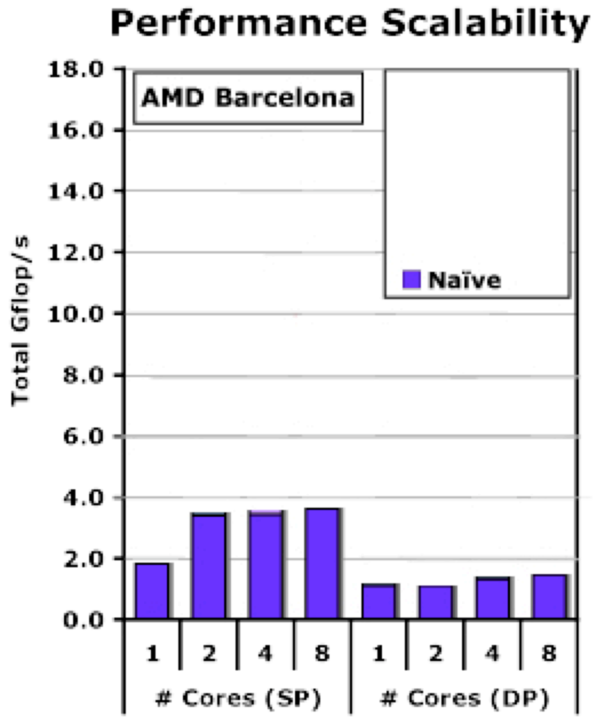
Solution: Auto-tuning

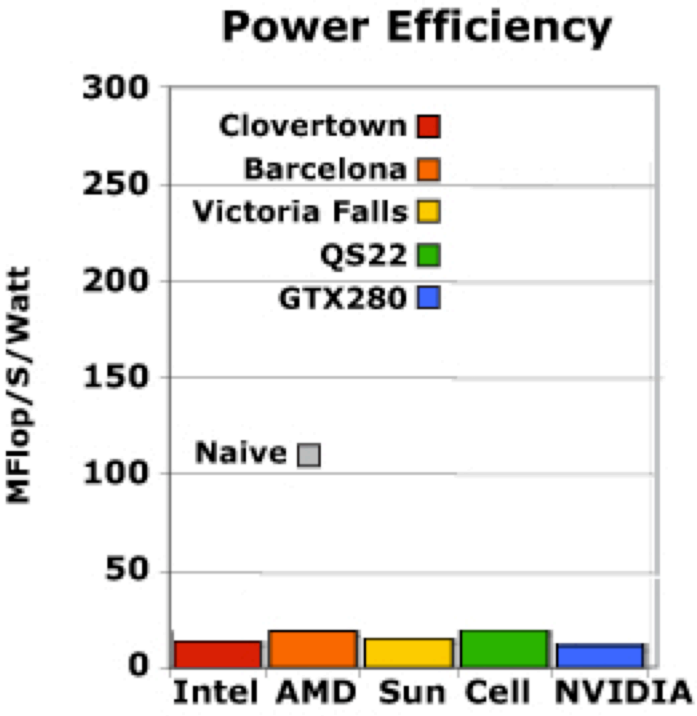
- Automate search across a complex optimization space
- Achieve performance far beyond current compilers
- Attain performance portability for diverse architectures


Identify domain-specific optimizations
Generate code variants based on these optimizations
Traverse parameter space for best configuration

Auto-Tuning for Finite Difference

- Attains performance portability across different multicore designs
- Only requires basic compiling technology
- Achieve serial performance, scalability, optimized power efficiency

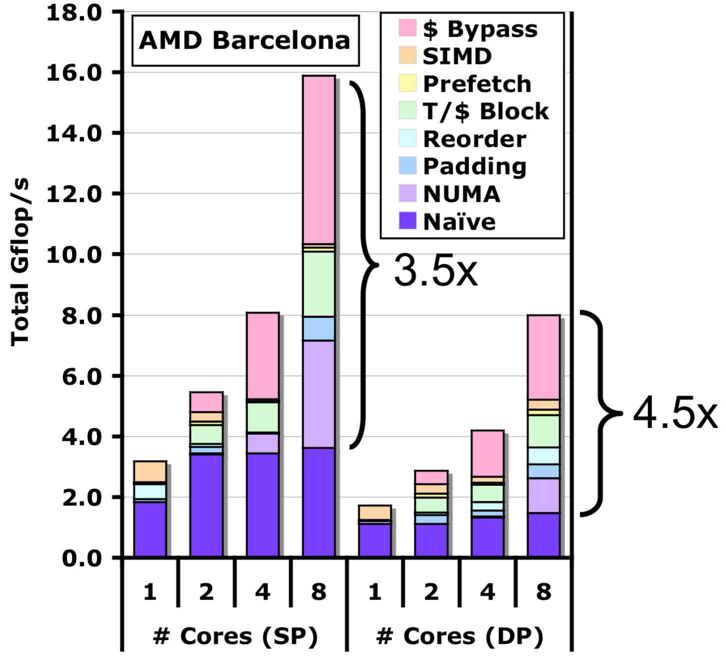




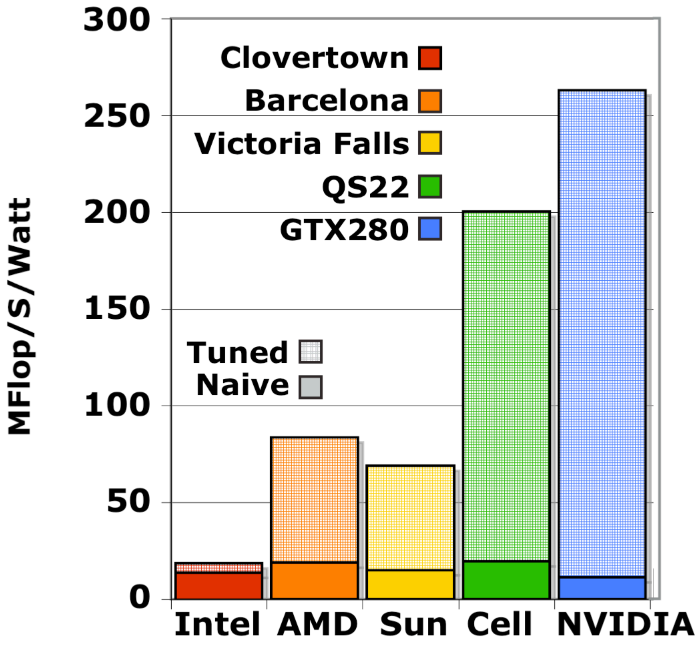
Auto-Tuning for Finite Difference

- Attains performance portability across different multicore designs
- Only requires basic compiling technology
- Achieve serial performance, scalability, optimized power efficiency

Performance Scalability

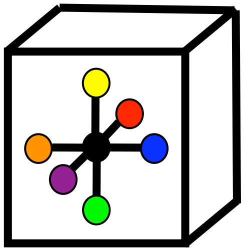


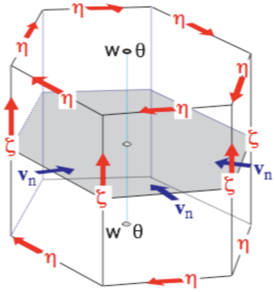
Power Efficiency

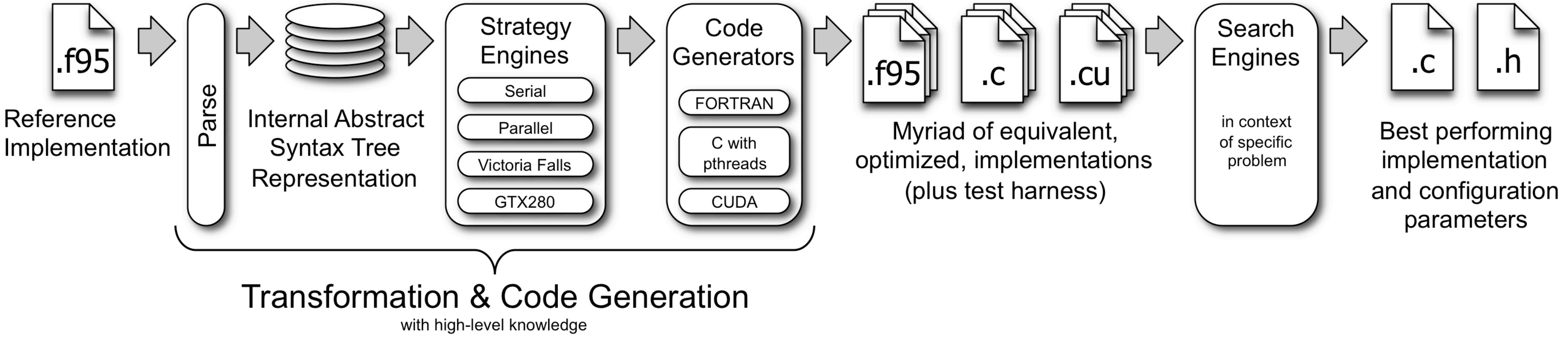


Generalized Stencil Auto-tuning Framework

- **Ability to tune many stencil-like kernels**
  - No need to write kernel-specific perl scripts
  - Uses semantic information from existing Fortran
- **Target multiple architectures**
  - Search over many optimizations for each architecture
  - Currently supports multi/manycore, GPUs
- **Better performance = Better energy efficiency**







.f95
Reference Implementation

Parse

Internal Abstract Syntax Tree Representation

Strategy Engines
- Serial
- Parallel
- Victoria Falls
- GTX280

Code Generators
- FORTRAN
- C with pthreads
- CUDA

.f95 .c .cu
Myriad of equivalent, optimized, implementations (plus test harness)

Search Engines
in context of specific problem

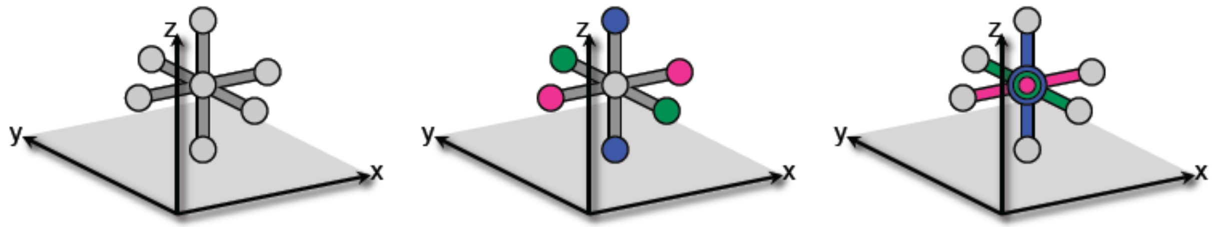
.c .h
Best performing implementation and configuration parameters

Transformation & Code Generation
with high-level knowledge

NERSC
NATIONAL ENERGY RESEARCH
SCIENTIFIC COMPUTING CENTER

Multi-Targeted Auto-Tuning

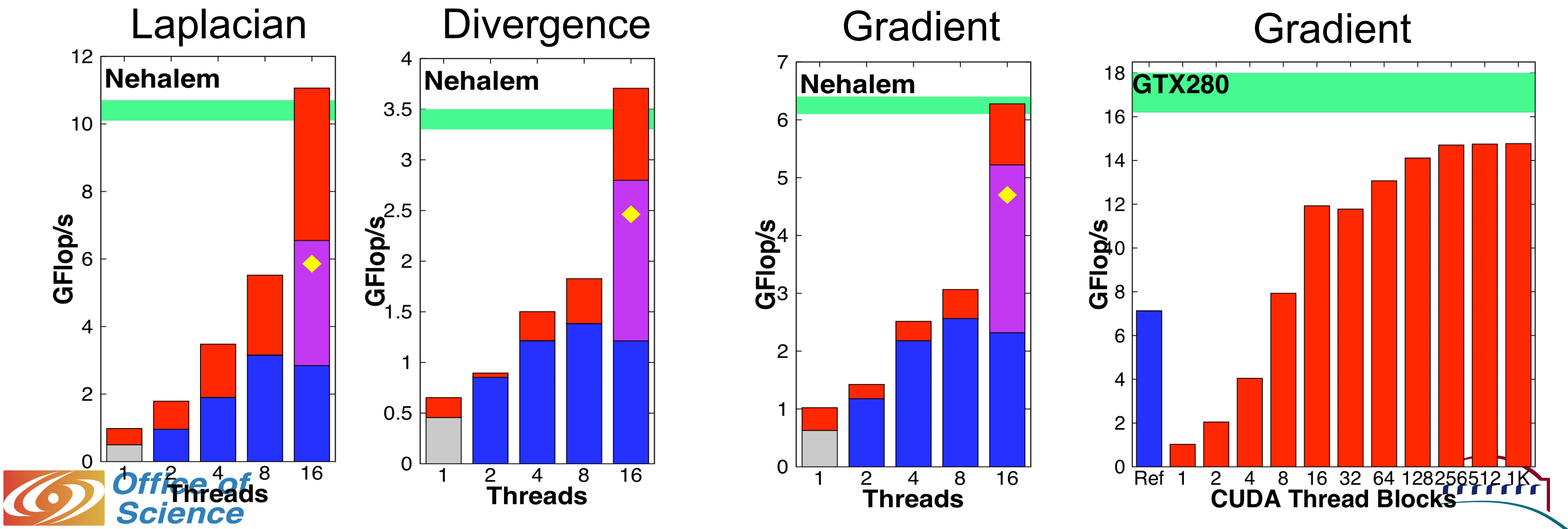
For Performance Portability



```
do k=2,nz-1,1
do j=2,ny-1,1
do i=2,nx-1,1

 uNext(i,j,k)=
  alpha*u(i,j,k)+
  beta*(u(i+1,j,k)+u(i-1,j,k)+
        u(i,j+1,k)+u(i,j-1,k)+
        u(i,j,k+1)+u(i,j,k-1)
       )
enddo
enddo
enddo
```

```
do k=2,nz-1,1
do j=2,ny-1,1
do i=2,nx-1,1

 u(i,j,k)=
  alpha*( x(i+1,j,k)-x(i-1,j,k) )+
   beta*( y(i,j+1,k)-y(i,j-1,k) )+
  gamma*( z(i,j,k+1)-z(i,j,k-1) )

enddo
enddo
enddo
```

```
do k=2,nz-1,1
do j=2,ny-1,1
do i=2,nx-1,1

 x(i,j,k)=alpha*( u(i+1,j,k)-u(i-1,j,k) )
 y(i,j,k)= beta*( u(i,j+1,k)-u(i,j-1,k) )
 z(i,j,k)=gamma*( u(i,j,k+1)-u(i,j,k-1) )

enddo
enddo
enddo
```



Office of Science
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

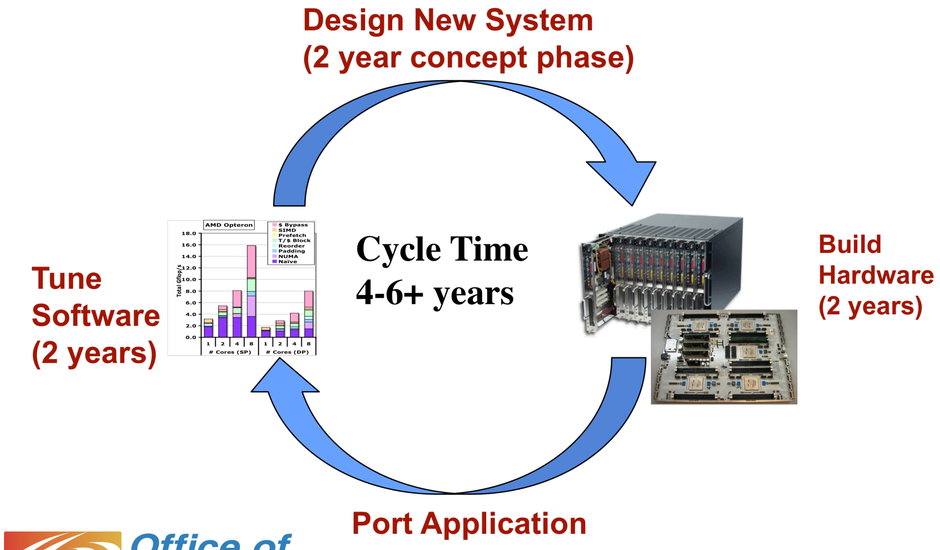
Rapid Prototyping of System Design

Using RAMP to Accelerate the hardware/software co-design cycle

Advanced Hardware Simulation (RAMP)

Enabling Hardware/Software/Science Co-Design

- Research Accelerator for Multi-Processors (RAMP)
  - Simulate hardware *before* it is built!
  - Break slow feedback loop for system designs
  - Enables tightly coupled hardware/software/science co-design *(not possible using conventional approach)*



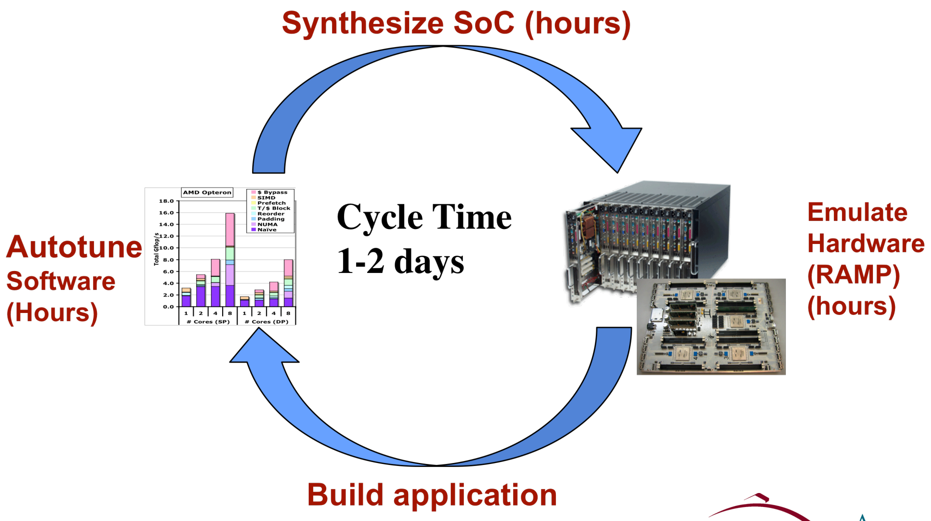
Design New System (2 year concept phase)

Build Hardware (2 years)

Port Application

Tune Software (2 years)

Cycle Time 4-6+ years



Synthesize SoC (hours)

Emulate Hardware (RAMP) (hours)

Build application

Autotune Software (Hours)

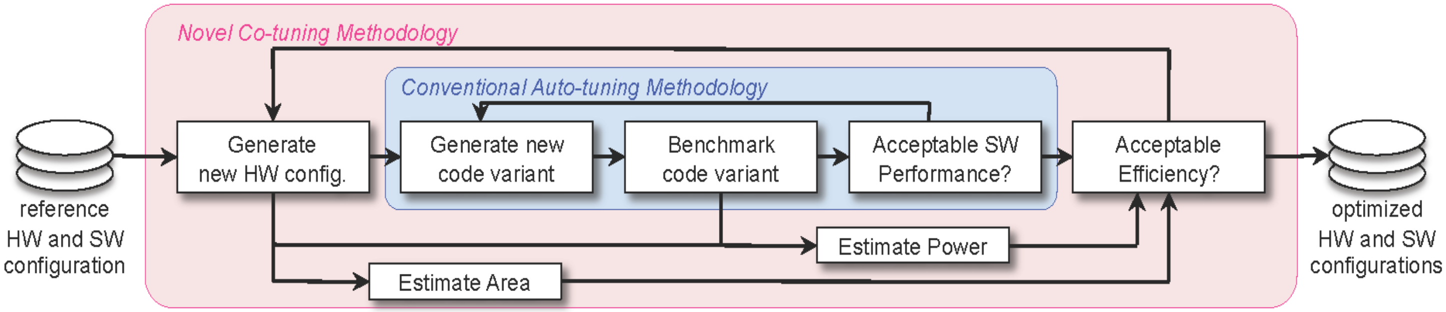
Cycle Time 1-2 days

Tuning Hardware to Fit the Problem

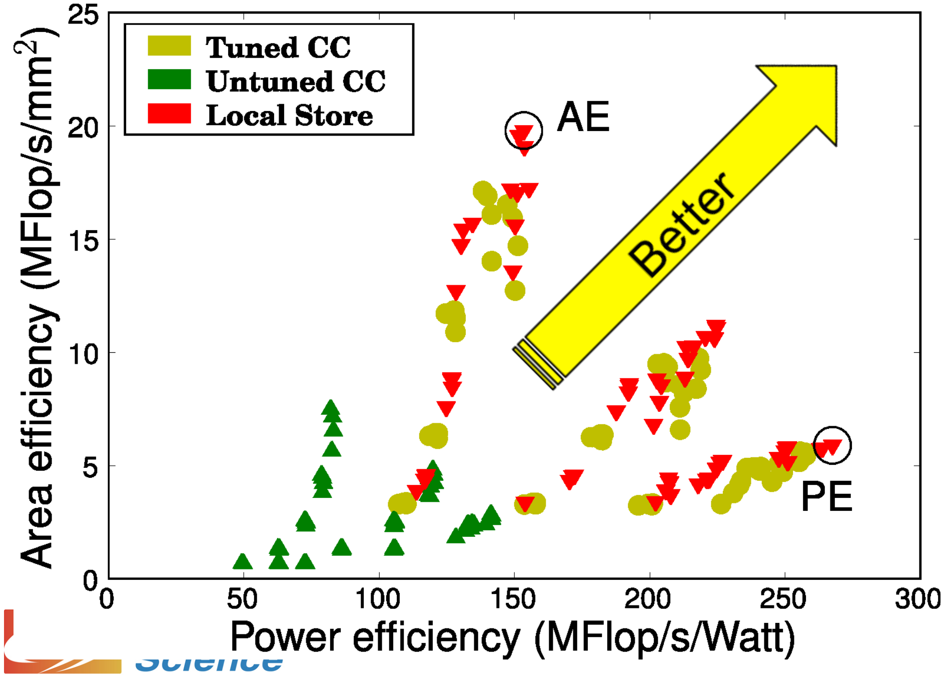
- Software Design Space Exploration: “auto-tuning”
  - Auto-search through parameter space of code optimizations
  - Tune to diverse & complex hardware
- Hardware Design Space Exploration:
  - What if hardware configuration was also parameterized?
  - Search through diverse space of hardware configurations
- What if you could do both together?
  - Auto-tune software for hardware
  - Auto-tune hardware for software
  - Repeat?
- Hardware/Software co-design
  - Demonstrate how to apply to HPC
  - Enable Energy Efficient computing for Extreme Scale Science

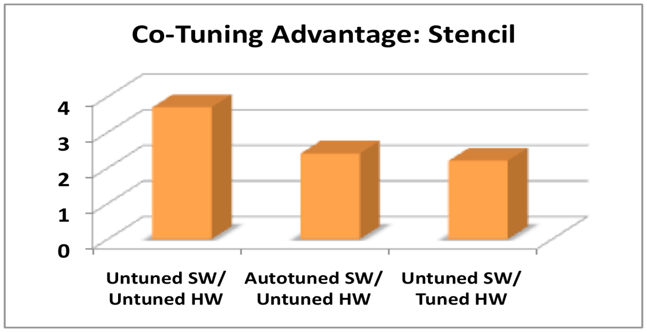
Hardware/Software Co-Design for Energy Efficiency

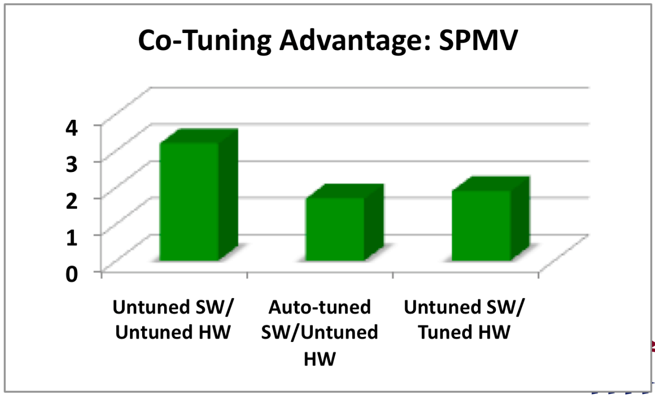
The approach: *Use auto-tuned code when evaluating architecture design points*


Novel Co-tuning Methodology
Conventional Auto-tuning Methodology
reference HW and SW configuration
Generate new HW config.
Generate new code variant
Benchmark code variant
Acceptable SW Performance?
Acceptable Efficiency?
Estimate Power
Estimate Area
optimized HW and SW configurations

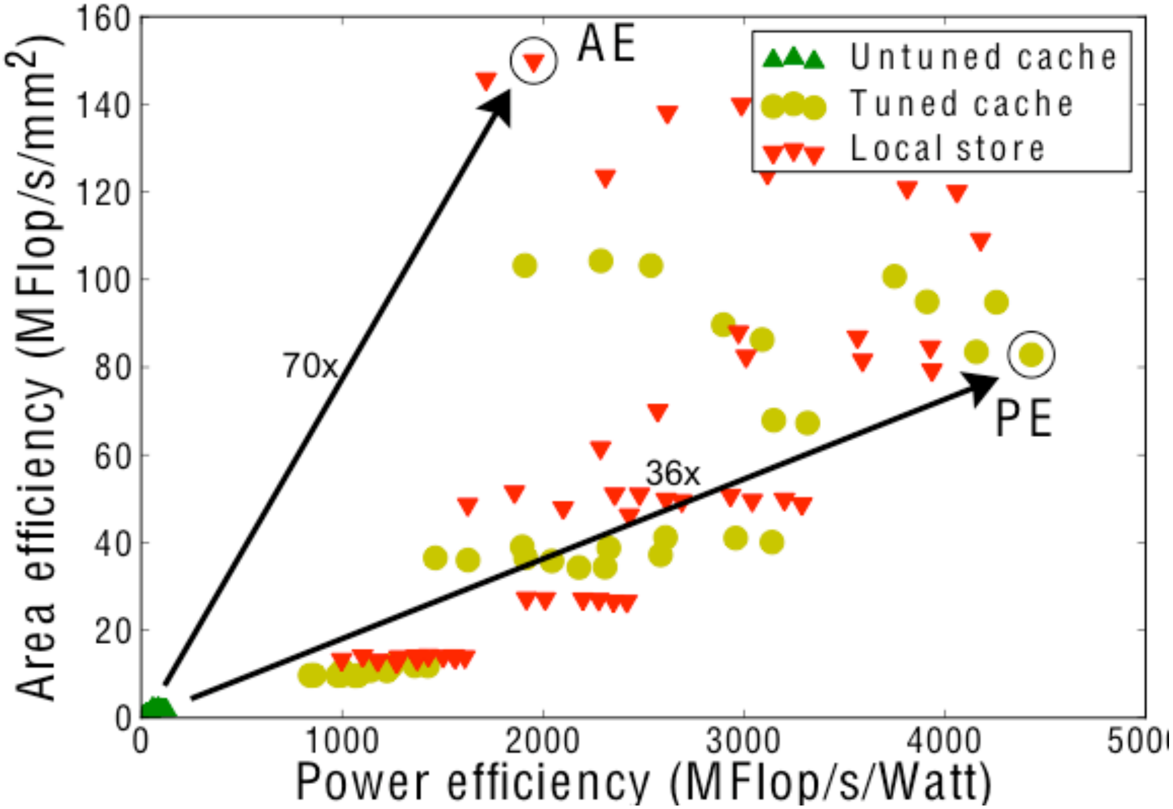
Co-Design can improve power-efficiency and area-efficiency by ~4x




Co-Tuning Advantage: Stencil

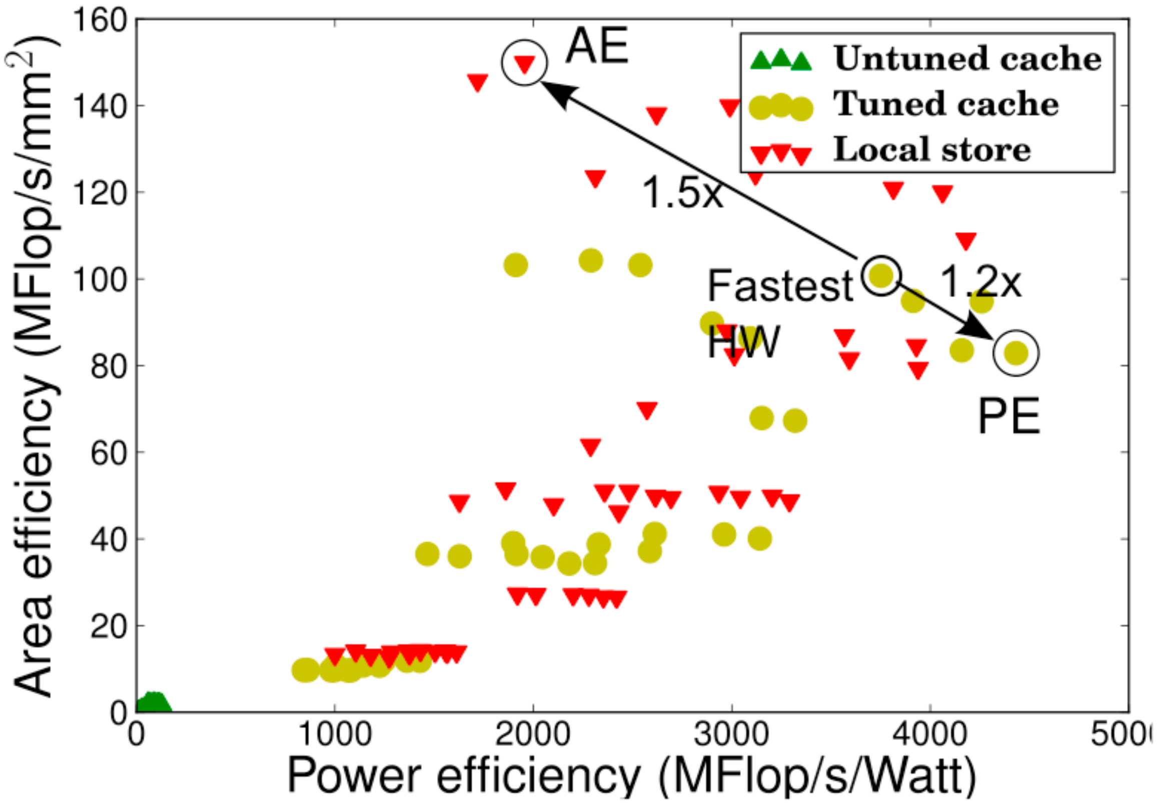

Co-Tuning Advantage: SPMV

GEMM Co-Design Results



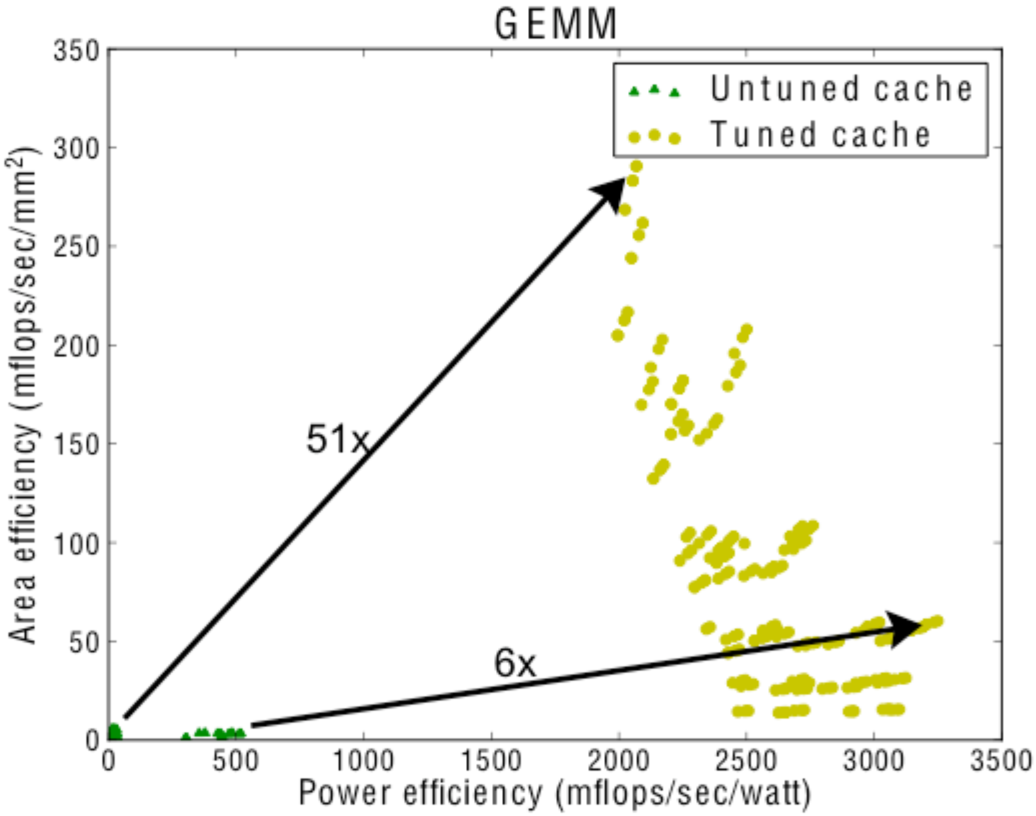
- Each point represents HW design point
  - Best SW performance chosen by autotuner
  - 72 unique configs
  - Runtime: 1 week

GEMM Co-Design Results



- Each point represents HW design point
  - Best SW performance chosen by autotuner
  - 72 unique configs
  - Runtime: 1 week

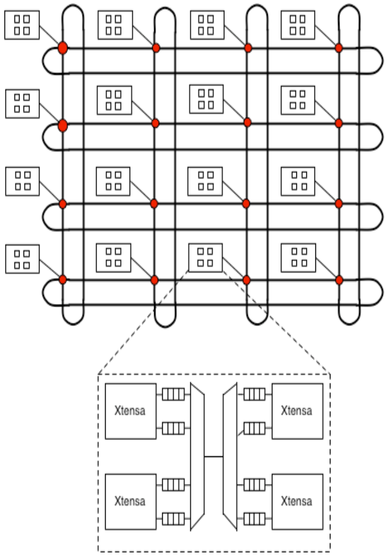
GEMM Co-Design Results

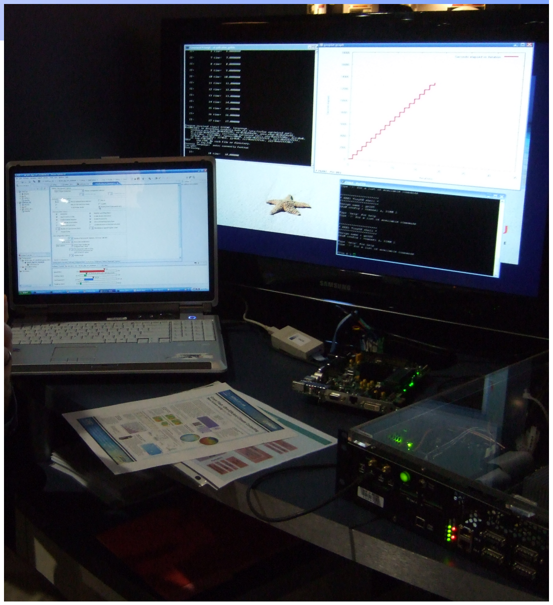


- **Generated through FPGA Emulation Flow**
  - **216 Unique Configs**
  - **Runtime: 1 day**
  - **125x speedup**

SC09 Green Flash Hardware Demo

- **Demonstrated during SC ‘09**
- **CSU atmospheric model ported to Tensilica Architecture**
  - **Dual Core Tensilica processors running atmospheric model at 25MHz**
  - **MPI Routines ported to custom Tensilica Interconnect**
- **Memory and PC Stats can be extracted for performance measurement**
- **Emulation performance advantage**
  - **Processor running at 25MHz vs. Functional model at 100 kHz**
  - ***250x Speedup***
- ***Actual code running - not representative benchmark***





Lets Put it All Together!

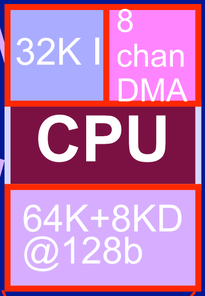
Strawman Design

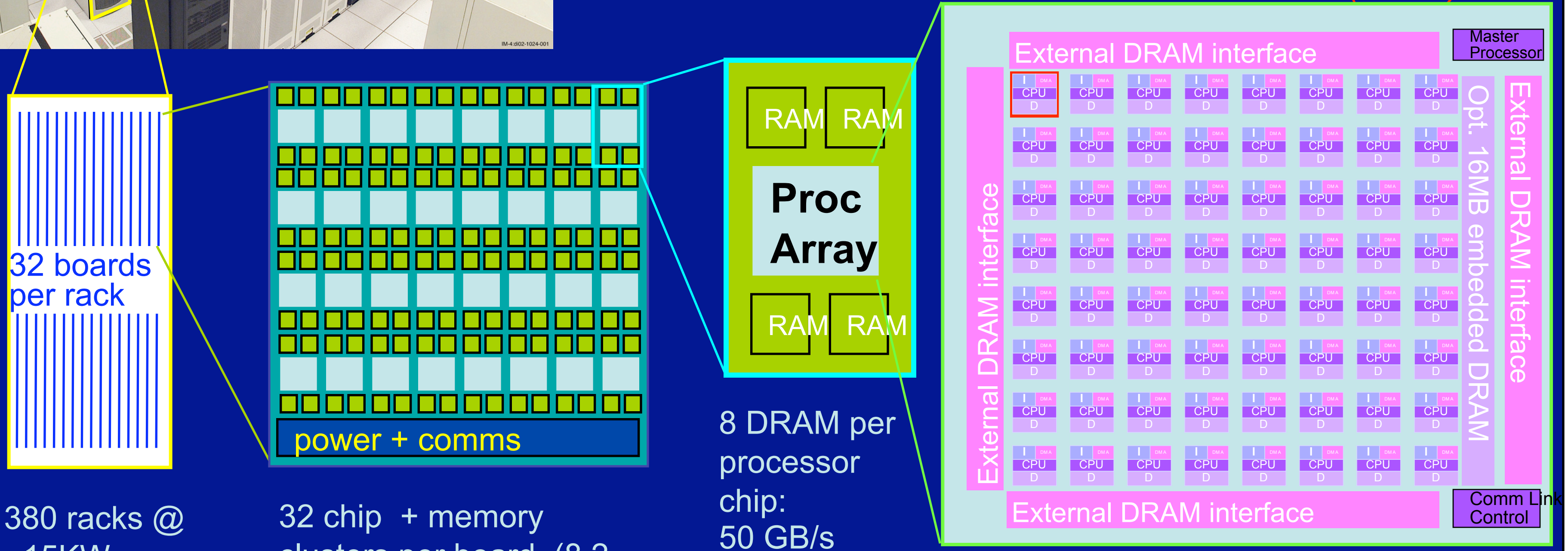
Climate Modeling System

Strawman 200PF Design

VLIW CPU:

- 128b load-store + 2 DP MUL/ADD + integer op/ DMA per cycle:
- Synthesizable at 1GHz Hz in commodity 45nm
- 0.5mm^2 core, 1.7mm^2 with inst cache, data cache data RAM, DMA interface, 0.15mW/MHz
- Double precision SIMD FP : 4 ops/cycle (4 GFLOPs)
- Vectorizing compiler, lightweight communications library, cycle-accurate simulator, debugger GUI
- 8 channel DMA for streaming from on/off chip DRAM
- Nearest neighbor 2D communications grid





380 racks @ ~15KW

32 chip + memory clusters per board (8.2 TFLOPS @ 450W

8 DRAM per processor chip: 50 GB/s

64 processors per 45nm chip 512 GFLOPS @ 10W

Green Flash Strawman System Design

We examined three different approaches (in 2008 technology)

- **AMD Opteron:** Commodity approach, lower efficiency for scientific codes offset by advantages of mass market. Constrained by legacy/binary compatibility.
- **BlueGene:** Generic embedded processor core and customize system-on-chip (SoC) to improve power efficiency for scientific applications
- **Tensilica XTensa:** Customized embedded CPU w/SoC provides further power efficiency benefits but maintains programmability.

  Mainstream design process, tool chain, commodity IP

Processor	Clock	Peak/ Core (Gflops)	Cores/ Socket	Sockets	Cores	Power
AMD Opteron	2.8GHz	5.6	2	890K	1.7M	1180 MW
IBM BG/P	850MHz	3.4	4	740K	3.0M	100 MW
Green Flash / Tensilica XTensa	650MHz	2.7	32	120K	4.0M	5 MW

NERSC
NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

Summary

- **Power is leading design constraint for future HPC**
  - **Future technology driven by handheld space**
  - **Notion of "commodity" moving <u>on-chip</u>**
- **Approach for Power Efficient HPC**
  - **Choose the science target first** *(climate in this case)*
  - **Design systems for applications** *(rather than the reverse)*
  - **Design hardware, software, scientific algorithms together using hardware emulation and auto-tuning**

  - ***<u>This is the right way to design efficient HPC systems!</u>***

Acknowledgements

UC Berkeley Students

- Marghoob Mohiyuddin
- Shoaib Kamil
- Jens Krueger
- Kaushik Datta
- Kamesh Madurri
- Cy Chan

Companies

- Tensilica Inc.
- Cray Inc.

LBNL Staff

- David Donofrio
- Leonid Oliker
- Michael Wehner
- Tony Drummond
- Woo-Sun Yang
- Norman Miller
- Sam Williams
- Chuck McParland

More Info

- Green Flash
  - http://www.lbl.gov/CS/html/greenflash.html
- NERSC System Architecture Group
  - http://www.nersc.gov/projects/SDSA
- The Berkeley View/Parlab
  - http://view.eecs.berkeley.edu
  - http://parlab.eecs.berkeley.edu/
- LBNL Future Technologies Group
  - http://crd.lbl.gov/ftg

NATIONAL ENERGY RESEARCH
SCIENTIFIC COMPUTING CENTER

U.S. DEPARTMENT OF ENERGY